# Guía de Estudio: LPIC DevOps Tools - Enfócate y Piensa

## 1. Control de versiones con Git

**- ¿Sabes cómo traer los cambios del repositorio remoto a tu entorno local?**

Para traer cambios del repositorio remoto a tu entorno local, se usa: *git pull*

**- ¿Que comando se usa para crear una nueva rama? ¿Y cuál para fusionarla?**

Para crear una nueva rama, usas: *git branch <nombre_rama>*

Para fusionarla, usas: *git merge <nombre_rama>*

**- ¿Como verificarías el estado actual de tus archivos antes de confirmar cambios?**

Para verificar el estado actual de tus archivos, se usa: *git status*

**- ¿Que comando usas cuando ya confirmaste tus cambios y quieres compartirlos con el resto del equipo?**

Para compartir los cambios don el resto del equipo después de confirmarlos, usas: *git push*

## 2. APIs y Métodos HTTP

**- ¿Cuál es el método HTTP que usarías para simplemente recuperar datos?**

El método HTTP para recuperar datos es: *GET*

**- Si quisieras crear un recurso nuevo, ¿cuál sería el método apropiado?**

Para crear un nuevo recurso nuevo, se usa: *POST*

**- ¿Que método se usaría si necesitas reemplazar completamente un recurso?**

El método para reemplazar completamente un recurso es: *PUT*

## 3. Arquitecturas y Despliegues

**- ¿Puedes distinguir entre arquitectura monolítica y de microservicios?**

La arquitectura monolítica es un diseño donde todos los componentes están integrados, mientras que, en microservicios, los componentes están distribuidos y son independientes.

**- ¿Como funciona un “Canary Release”? ¿Y un despliegue “Blue/Green”?**

El “Canary Release” lanza una nueva versión a un pequeño grupo de usuarios antes de hacerla disponible a todos. El despliegue “Blue/Green” tiene dos entornos, uno en producción y otro para probar la nueva versión.

**- ¿Qué tipo de despliegue permite lanzar nuevas versiones a un subconjunto de usuarios antes de exponerla a todos?**

El tipo de despliegue que permite lanzar versiones a un subconjunto de usuarios es el “Canary Release”.

**- ¿Con que tipo de estrategia cambiarias todo el tráfico a una nueva versión validada?**

Para cambiar todo el tráfico a una nueva versión validada, usarías una estrategia “Blue/Green”.

## 4. Docker y Contenedores

**- ¿Que archivo define como se construye una imagen Docker?**

El archivo que define cómo se construye una imagen Docker es el *Dockerfile*.

**- Si tienes varios contenedores que necesitan ejecutarse como parte de una misma aplicación, ¿Qué herramienta utilizarías?**

Para ejecutar varios contenedores como parte de una aplicación, usarías: Docker Compose.

**- ¿Dónde se almacenan y comparten comúnmente las imágenes Docker?**

Las imágenes Docker se almacenan y comparten comúnmente en Docker Hub.

**- ¿Por qué los contenedores suelen ser más ligeros y rápidos que las máquinas virtuales?**

Los contenedores suelen ser más ligeros y rápidos que las máquinas virtuales porque comparten el mismo kernel del sistema operativo en lugar de virtualizar uno completo.

## 5. Automatización y Configuración

**- ¿Que herramienta te permite aprovisionar automáticamente maquinas creadas con Vagrant?**

La herramienta que permite aprovisionar máquinas creadas con Vagrant es: Puppet o Chef (dependiendo de la configuración).

**- ¿Sabes cuál es la diferencia entre una “tarea” y un “módulo” en Ansible?**

En Ansible, una “tarea” es una acción a ejecutar y un “módulo” es el bloque de código que realiza una tarea específica.

**- ¿Dónde defines que máquinas va a gestionar Ansible?**

Las máquinas que va a gestionar Ansible se definen en el archivo de inventario.

**- ¿Que formato de archivo se usa habitualmente para escribir Playbooks?**

El formato de archivo que se usa habitualmente para escribir Playbooks es: YAML.

## 6. Infraestructura como Código y Virtualización

**- ¿Qué herramienta te ayuda a crear entornos portátiles para desarrollo?**

La herramienta que te ayuda a crear entornos portátiles para desarrollo es: Vagrant.

**- ¿Qué herramienta permite construir imágenes de máquinas (para AWS, por ejemplo) de forma automática a partir de una plantilla?**

La herramienta que permite construir imágenes de máquinas automáticamente es: Packer.

**- ¿Qué diferencia hay entre usar Vagrant y usar Packer?**

La diferencia entre usar Vagrant y usar Packer es que Vagrant se enfoca en la gestión de entornos virtualizados, mientras que Packer se utiliza para crear imágenes de máquinas listas para ser desplegadas.

## 7. Nube y Servicios

**- ¿Conoces las diferencias entre SaaS, PaaS, IaaS y FaaS?**

- **SaaS:** Software como servicio. Todo hecho por el proveedor. Solo lo usas.
- **PaaS:** Plataforma como servicio. Tú construyes la app, el proveedor da el entorno.
- **IaaS:** Infraestructura como servicio. Tú montas todo, pero no te preocupas por el hardware.
- **FaaS:** Función como servicio. Tú escribes funciones que se ejecutarán cuando se necesiten.

**- ¿Qué modelo te da acceso directo a recursos como almacenamiento o servidores?**

El modelo que da acceso directo a recursos como almacenamiento o servidores es: IaaS.

**- ¿Que es OpenStack y que papel cumple en entornos de nube?**

OpenStack es una plataforma de código abierto para crear y gestionar nubes privadas y públicas.

## 8. CI/CD y Automatización de Workflows

**- ¿Que significan las siglas CI/CD y que procesos implican?**

Significa Integración continua / Entrega continua y abarca los procesos de integración de código y despliegue automatizado.

**- ¿Que herramienta muy conocida usa “pipelines” definidos como código para automatizar tareas como pruebas y despliegues?**

La herramienta muy conocida que usa “pipelines” definidos como código es: Jenkins.

**- ¿Por qué la automatización de pruebas es crucial en un pipeline?**

La automatización de pruebas es crucial en un pipeline porque garantiza que el código esté funcionando correctamente antes de ser desplegado, evitando errores en producción.

## 9. Gestión de Configuración

**- ¿Qué hacen herramientas como Chef, Puppet o Ansible?**

Son herramientas de automatización de configuración y gestión de infraestructura.

**- ¿Sirven para monitorear sistemas, o para configurar y mantener su estado?**

Su principal propósito es configurar, desplegar y mantener el estado deseado de los sistemas (servidores, servicios, redes, etc.) no tanto el monitorio (aunque pueden complementar ese proceso).

**- ¿Cómo garantizan estas herramientas que un servidor se mantenga con una configuración especifica?**

Estas herramientas garantizan que un servidor mantenga una configuración específica mediante el uso de “roles”, “playbooks” y otras configuraciones declarativas.

## 10. Observabilidad

**- En el stack ELK, ¿qué componente se usa para visualizar los datos?**

En el stack ELK, el componente usado para visualizar los datos es: Kibana.

**- ¿Cual recoge y transforma datos, y cual los almacena e indexa?**

Logstash recoge y transforma los datos, mientras que Elasticsearch los almacena e indexa.

**- ¿Que función cumplen los "Beats"?**

Los "Beats" son agentes ligeros que envían datos a Logstash o Elasticsearch.

## 11. Kubernetes

**- ¿Cuál es el objeto más básico y fundamental de Kubernetes?**

El objeto más básico y fundamental de Kubernetes es el Pod.

**- ¿Que estructura agrupa uno o más contenedores que comparten recursos como red y almacenamiento?**

La estructura que agrupa uno o más contenedores que comparte recursos como red y almacenamiento es el Pod.

**- ¿Como se define un servicio que expone aplicaciones en un clúster?**

En Kubernetes, un Servicio (Service) es un recurso que define una abstracción de red que expone una o más aplicaciones (pods) para que puedan ser accedidas dentro o fuera del clúster, de forma estable y predecible, incluso cuando los pods subyacentes cambien.