TaiLabs

OPOSICIONES TAI · ADMINISTRACIÓN GENERAL DEL ESTADO

Bloque IV · Tema IV.03

Administración de Servidores de Correo Electrónico y sus Protocolos. Administración de Contenedores y Microservicios

GUÍA INTENSIVA DE ESTUDIO · ORIENTADA A EXAMEN

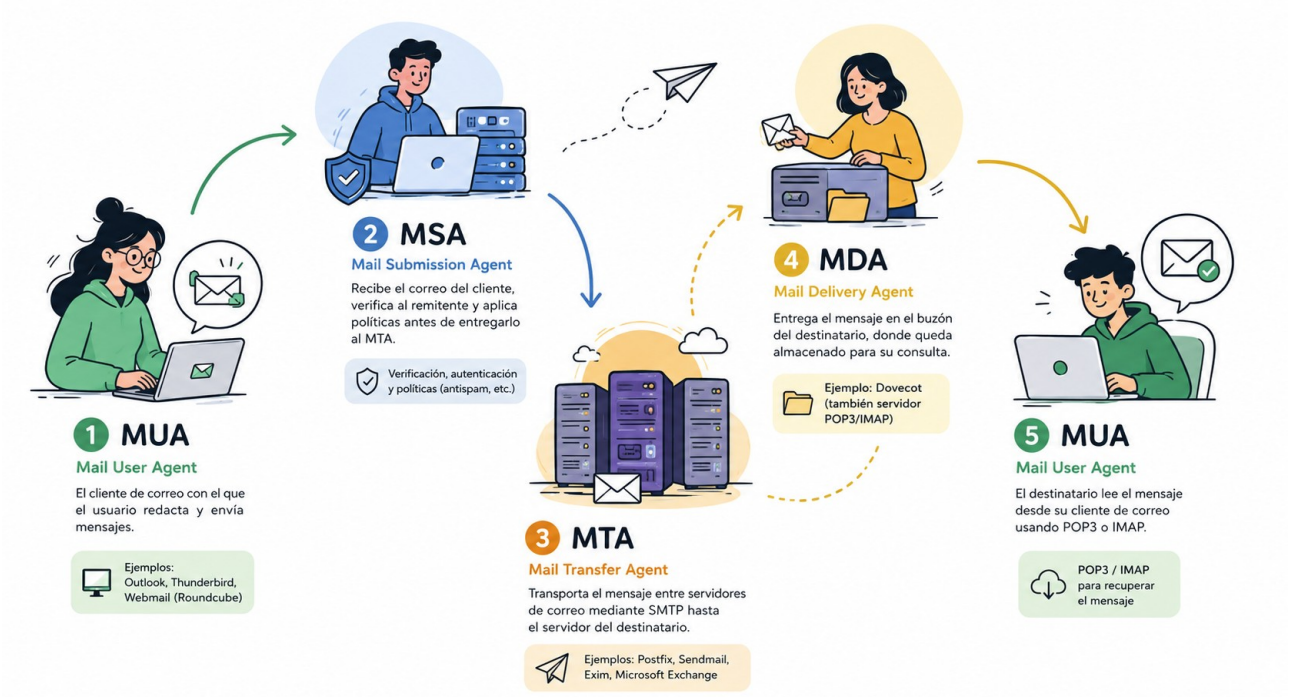
Contenido actualizado a agosto 2026 · Preguntas 2016–2026

1. El correo electrónico: arquitectura y funcionamiento

1.1 Componentes del sistema de correo

El correo electrónico funciona como una cadena de agentes, cada uno con un papel definido:

- **MUA** (*Mail User Agent*): el **cliente de correo** con el que el usuario redacta y lee mensajes. Ejemplos: Outlook, Thunderbird, webmail (Roundcube).
- **MSA** *(Mail Submission Agent)*: el agente de software que **recibe y verifica los correos** antes de entregarlos al MTA.
- **MTA** (*Mail Transfer Agent*): el **servidor de correo** que transporta los mensajes entre servidores mediante SMTP. Ejemplos: **Postfix**, Sendmail, Exim, Microsoft Exchange.
- **MDA** (*Mail Delivery Agent*): entrega el mensaje en el buzón del destinatario, desde donde se recupera con POP3 o IMAP. Ejemplo: **Dovecot** (que además hace de servidor POP3/IMAP).



Es importante no confundir cliente y protocolo: **Outlook es un cliente** (MUA), no un protocolo. Los protocolos de correo son SMTP, POP3 e IMAP.

> ⚠️ **CLAVE DE EXAMEN · 2022 · P66**
> «¿Cuál NO es un protocolo de correo?» → **Outlook**: es un cliente de correo. IMAP, SMTP y POP3 sí son protocolos (recepción, envío y descarga, respectivamente).

1.2 El viaje de un correo: qué protocolo actúa en cada tramo

El recorrido de un mensaje entre dos usuarios tiene tres tramos: los dos primeros usan SMTP y el tercero usa POP3 o IMAP:

- Remitente → su servidor: usa el protocolo **SMTP** (el cliente entrega el mensaje).

- Servidor de origen → servidor de destino: usa el protocolo **SMTP** de nuevo (el transporte entre servidores siempre es SMTP).
- Servidor de destino → destinatario: usa el protocolo **POP3 o IMAP** (el destinatario recupera el mensaje de su buzón cuando se conecta).

⚠ **CLAVE DE EXAMEN · 2019 · P62**

SMTP se usa en el envío (cliente→servidor de origen y servidor→servidor) y **POP3/IMAP en la recuperación** (del servidor de destino al ordenador del destinatario). Los distractores invierten los papeles poniendo POP3/IMAP a enviar o SMTP a entregar al destinatario.

2. Protocolos de correo electrónico

2.1 SMTP: el protocolo de envío

SMTP (*Simple Mail Transfer Protocol*) gestiona el **correo saliente**: es el protocolo de envío. Funciona como un diálogo de comandos de texto entre cliente y servidor:

Comando SMTP	Función
HELO / EHLO	Saludo inicial e identificación del cliente (EHLO en SMTP extendido).
MAIL FROM:	Indica el remitente del mensaje.
RCPT TO:	Indica cada destinatario.
DATA	Comienza el contenido del mensaje (termina con una línea con solo un punto).
RSET	Aborta la transacción en curso.
VRFY	Pide verificar si una dirección existe.
QUIT	Cierra la sesión.

El servidor responde con **códigos numéricos de tres cifras**: la primera cifra indica el resultado (2xx éxito, 3xx acción pendiente, 4xx error temporal, 5xx error permanente):

- **220**: servicio listo (al establecer la conexión).
- **250**: acción completada correctamente — es el código con el que acepta el **final de la transmisión** del mensaje.
- **354**: «inicie la entrada del mensaje» (respuesta a DATA).
- **450**: buzón no disponible, error temporal.
- **550**: buzón inexistente, error permanente.

⚠ **CLAVE DE EXAMEN · 2016 · P10 · 2017 · P79**

SMTP gestiona el correo saliente — descargar los mensajes al ordenador es cosa de POP3, no de SMTP. IMAP obtiene los emails, pero no los descarga localmente (2016 · P10). El código que indica que **la transmisión del mensaje ha finalizado** correctamente es el **250** (220 = servicio listo; 450 = buzón no disponible; 200 no existe en SMTP) (2017 · P79).

2.2 POP3: descarga del buzón

POP3 (*Post Office Protocol v3*) **descarga los mensajes al equipo** del usuario y, **en su configuración tradicional** la mayoría de clientes, **los eliminaba del servidor** por defecto. Su sincronización es **unidireccional** (servidor → cliente): lo que el usuario organice en carpetas en un dispositivo **no** se replica en los demás. Sus comandos principales:

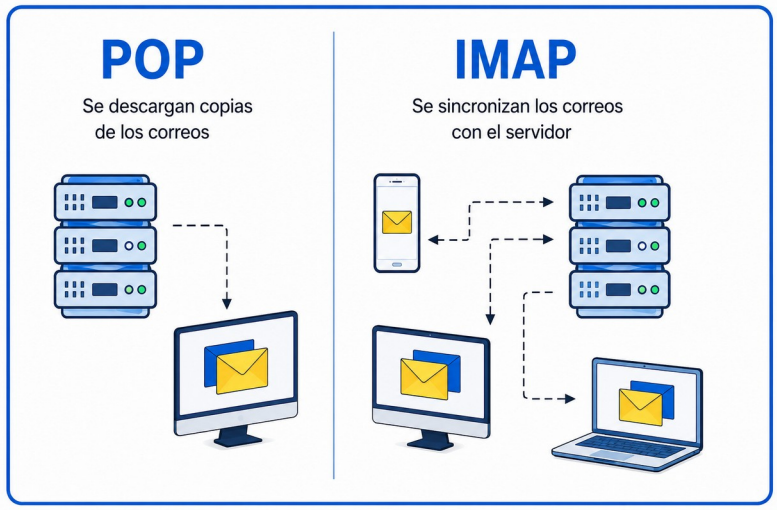
Comando POP3	Función
USER / PASS	Autenticación: usuario y contraseña.
STAT	Número de mensajes del buzón y tamaño total en bytes.
LIST	Lista los mensajes con su tamaño individual.
RETR n	Descarga (recupera) el mensaje n.
DELE n	Marca el mensaje n para borrado.
RSET	Deshace las marcas de borrado.
TOP	Muestra la cabecera de un mensaje.
QUIT	Cierra la sesión (y hace efectivos los borrados).

⚠️ **CLAVE DE EXAMEN · 2018 · P79**

El comando POP3 que devuelve el **número de mensajes y su tamaño total en bytes** es **STAT**. RETR descarga un mensaje concreto y PASS envía la contraseña.

2.3 IMAP: el buzón vive en el servidor

IMAP (Internet Message Access Protocol) mantiene los mensajes almacenados en el servidor: el usuario los visualiza y organiza sin descargarlos, y puede iniciar sesión desde varios clientes a la vez viendo siempre lo mismo (sincronización bidireccional, incluidas las carpetas). El correo permanece en el servidor hasta que el usuario lo elimina permanentemente. Es el protocolo natural cuando se accede desde varios dispositivos; POP3 solo compensa si se quiere conservar el correo localmente y liberar el servidor.



Comando IMAP	Función
LOGIN	Autenticación.
LIST	Muestra las carpetas disponibles.
SELECT	Abre una carpeta de correo en el servidor.
FETCH	Recupera un mensaje o parte de él.
STORE	Modifica las marcas de un mensaje: \Seen, \ Answered, \Flagged (importante), \Deleted (marcado para eliminar) o \Draft.
COPY	Copia un mensaje a otra carpeta.
EXPUNGE	Elimina permanentemente los mensajes marcados como eliminados
LOGOUT	Finaliza la sesión con el servidor

⚠ CLAVE DE EXAMEN · 2024 · P63 · 2024 · P18 (SP)

Ventaja de IMAP sobre POP3: **permite visualizar los correos directamente en el servidor sin descargarlos** (2024 · P63). La afirmación INCORRECTA típica: atribuir a **POP3** la réplica de la organización en carpetas entre dispositivos — eso es de IMAP; POP3 solo sincroniza en un sentido, descargando (2024 · P18 (SP)).

2.4 Puertos de los protocolos de correo

Cada protocolo tiene su puerto en claro y su puerto **seguro** (sobre SSL/TLS). Son de las preguntas más repetidas del tema:

Protocolo	Puerto en claro	Puerto seguro (SSL/TLS)
SMTP	25	465 (SMTPS) / (587 para envío autenticado con STARTTLS)
POP3	110	995 (POP3S)
IMAP	143	993 (IMAPS)
QMTP (Quick Mail Transfer Protocol)	209 (IANA)	—

⚠ CLAVE DE EXAMEN · 2016 · P35 · 2018 · P77 · 2026 · P17 (SP) · 2022 · P10 (SP)

IMAPS (IMAP sobre SSL/TLS) = puerto 993 sobre TCP, NO sobre UDP. Ha caído CUATRO veces: en 2016 con la trampa TCP/UDP (es TCP), en 2018 y 2026 a secas, y en 2022 pidiendo el trío seguro completo: **POP:995 / SMTP:465 / IMAP:993**. El 995 es POP3S: no confundirlos.

⚠ CLAVE DE EXAMEN · 2019 · P19 (SP) · 2016 · P6

SMTPS usa el puerto 465 — el 25 es SMTP sin cifrar y el 443 es HTTPS (2019 · P19 (SP)). Pregunta rebuscada de 2016: el puerto IANA de **QMTP es el 209** (2016 · P6).

💡 **TRUCO MNEMOTÉCNICO**

Puertos seguros del correo: **IMAPS 993 y POP3S 995** van seguidos — «la **I** va antes que la **P**, el **3** antes que el **5**». Y **SMTPS 465**: el que envía va por libre.

2.5 Correo seguro de extremo a extremo: S/MIME

Los puertos seguros cifran el transporte, pero no el mensaje en sí. **S/MIME** (*Secure/Multipurpose Internet Mail Extensions*) es el estándar para el envío de **correos electrónicos seguros**: añade **cifrado y firma digital** al propio mensaje, sobre los protocolos existentes (SMTP, POP3, IMAP), garantizando confidencialidad, integridad y autenticación. Los certificados X.509 aportan además el no repudio. La alternativa clásica sin certificados centralizados es **PGP/OpenPGP**.

⚠️ **CLAVE DE EXAMEN · 2016 · P44**

Protocolo para el envío de correos electrónicos **seguros**: **S/MIME**. Los distractores no tenían relación con el correo: LVM (volúmenes lógicos), VRRP (redundancia de routers) y Gluster (almacenamiento distribuido).

2.6 Exchange ActiveSync

Exchange ActiveSync (EAS) es el protocolo de **Microsoft** para que los dispositivos móviles sincronicen correo, calendario, contactos y tareas con servidores **Exchange**, incluso con acceso sin conexión. Sus características de examen:

- Basado en **HTTP y XML** (en su variante binaria comprimida WBXML) — **no usa JSON**.
- Admite **SSL/TLS** entre cliente y servidor.
- Permite el **Remote Wipe**: borrado remoto del contenido de un dispositivo móvil perdido o robado.

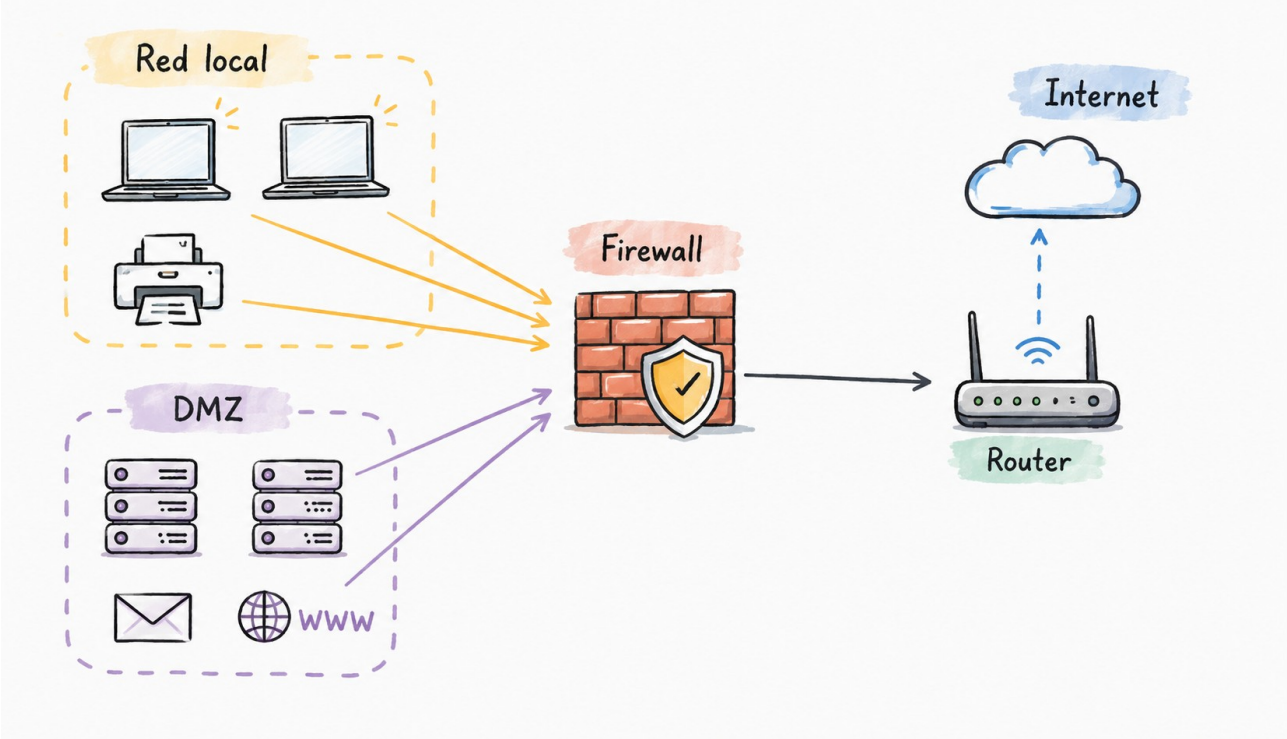
⚠️ **CLAVE DE EXAMEN · 2016 · P9 · 2016 · P52 (R) · 2018 · P78**

El protocolo de sincronización con Exchange **basado en HTTP y XML** y con acceso sin conexión es **Exchange ActiveSync** (2016 · P9); como tecnología general de sincronización PC-móvil de Microsoft, **ActiveSync** (2016 · P52 (R)). La afirmación FALSA sobre EAS: que use **JSON** para reducir el peso de los mensajes — usa XML/WBXML (2018 · P78).

3. Administración del servidor de correo

3.1 Ubicación en la red: la DMZ

El servidor de correo debe ser accesible desde Internet sin exponer la red interna. Por ello el **CCN-CERT** (equipo de respuesta a incidentes del Centro Criptológico Nacional) recomienda ubicarlo en una **zona desmilitarizada (DMZ)**: una red intermedia, separada por cortafuegos tanto de Internet como de la LAN, donde se sitúan los servicios expuestos al exterior (como servidores web o de correo).



La DMZ existe partiendo de la premisa de que alguno de los servidores va a ser comprometido tarde o temprano, porque son los únicos alcanzables desde Internet. Si un atacante toma el control del servidor de correo, lo único que ha ganado es un pie en una red aislada. La DMZ reduce la superficie desde la que el atacante podría acceder a la Red Local, ya que el cortafuegos interno impediría las conexiones iniciadas desde la DMZ. Un equipo de la Red Local sí puede comunicarse con los servidores de la DMZ, ya que la conexión la inician los equipos de la Red Local.

⚠ **CLAVE DE EXAMEN · 2016 · P8**

Según el CCN-CERT, el servidor de correo corporativo debe estar **ubicado en una zona desmilitarizada (DMZ)** — ni en la red interna (expondría la LAN), ni en la externa (sin protección), ni aislado (no daría servicio).

3.2 El DNS del correo: registro MX

La recepción de correo de un dominio se configura en su **DNS**: el registro **MX** (*Mail Exchange*) indica qué servidor recibe el correo del dominio (puede haber varios con distinta prioridad: el número menor gana) de forma que el servidor emisor sabe a dónde entregar el mensaje y en qué orden intentarlo.

Otros registros implicados: **A** resuelve el nombre del servidor a su IP y **TXT** almacena texto arbitrario — es donde se publican SPF, DKIM y DMARC que veremos a continuación.

⚠️ **CLAVE DE EXAMEN · 2018 · P21 (SP)**
Para configurar la **recepción** de correo de un dominio hace falta el registro DNS **MX** — el registro A resuelve nombres a IP, CNAME crea un alias y TXT guarda texto (SPF/DKIM), pero ninguno designa por sí mismo el servidor receptor.

3.3 Autenticación del remitente: SPF, DKIM y DMARC

Para combatir la suplantación de remitente (*spoofing*) y el spam, el dominio publica en su DNS tres mecanismos complementarios:

- **SPF** (*Sender Policy Framework*): registro (TXT) que **indica qué servidores están autorizados a enviar correo en nombre del dominio**. El receptor comprueba si la IP remitente está en la lista.
- **DKIM** (*DomainKeys Identified Mail*): el servidor emisor **firma criptográficamente** cada mensaje; la clave pública se publica en el DNS para verificar la firma.
- **DMARC** (*Domain-based Message Authentication, Reporting and Conformance*): pasa si SPF o DKIM verifican correctamente y, además, el dominio validado está alineado con el dominio de la cabecera From. **Basta con que uno de los dos cumpla ambas condiciones**. Si ninguno lo hace, DMARC falla y se aplica la política publicada en _dmarc.<dominio>: p=none (no solicita acción), p=quarantine (cuarentena) o p=reject (rechazo).

⚠️ **CLAVE DE EXAMEN · 2016 · P7 · 2018 · P35 (SP) (R)**
Los registros **SPF indican los servidores de correo autorizados a enviar en nombre de un dominio** (2016 · P7). Como medida de protección de servidores SMTP frente a ciberataques, la respuesta válida era **SPF** para evitar que servidores no autorizados envíen correo en nombre del dominio — Grafana es una herramienta de monitorización, y XiaoBa y Emotet son malware, no defensas (2018 · P35 (SP) (R)).

3.4 Postfix: configuración esencial

Postfix es el MTA libre más extendido en Linux. Sus dos ficheros de configuración viven en **/etc/postfix/**:

- **main.cf**: la configuración **global** del servidor (parámetros como myhostname, mydomain, mynetworks o el nombre anunciado en el saludo HELO/EHLO). Tras modificarlo hay que reiniciar o recargar el servicio.
- **master.cf**: define los **procesos y servicios** internos de Postfix (smtpd, pickup, qmgr...), no los parámetros del correo.

Utilidades de administración: **postconf** muestra la configuración activa, **mailq** (o postqueue -p) lista la cola de correo, **postqueue -f** fuerza el reintento de entrega y **postfix reload** recarga la configuración sin cortar el servicio.

⚠️ **CLAVE DE EXAMEN · 2024 · P21 (SP) (R)**
Cambiar el nombre que aparece en el saludo **HELO** de todos los servidores SMTP: editar el parámetro **$helo_name en /etc/postfix/main.cf** y reiniciar Postfix (así lo dio por válido el examen; en la documentación actual de Postfix el parámetro es smtp_helo_name). Las trampas: $client no controla el saludo, EHLO es un comando del diálogo SMTP (no configuración) y master.cf define

procesos y servicios internos.

4. Contenedores

4.1 Concepto y tecnologías

Un **contenedor** empaqueta una aplicación con todas sus dependencias (librerías, configuración) para ejecutarla aislada sobre el kernel del sistema anfitrión. A diferencia de una máquina virtual, **no incluye un sistema operativo completo**: comparte el kernel del host, por lo que arranca en segundos y consume muchos menos recursos.

Tecnologías de **contenerización** que hay que reconocer:

- **Docker**: la plataforma de contenedores más popular.
- **containerd** y **CRI-O**: motores de ejecución (*runtimes*) de contenedores.
- **Podman**: alternativa a Docker sin demonio central, de Red Hat.
- **LXC**: contenedores de sistema en Linux (los usa Proxmox VE).
- **Kubernetes** NO es un contenedor: es el **orquestador** que los gestiona (sección 5).

⚠️ **CLAVE DE EXAMEN · 2024 · P64**
Tecnologías relacionadas con la contenerización y gestión de contenedores: **Docker, Containerd y Podman**. Los distractores mezclaban nginx (servidor web) y Ubuntu/Debian (sistemas operativos).

4.2 Docker: arquitectura

Docker se articula en torno a cuatro conceptos:

- La **imagen**: plantilla inmutable con la aplicación y sus dependencias. Se construye a partir de un fichero de instrucciones llamado **Dockerfile** (con docker build).
- El **contenedor**: una instancia en ejecución de una imagen.
- El **registro** (*registry*): almacén de imágenes; el público por defecto es **Docker Hub**.
- El **demonio** dockerd, que gestiona imágenes y contenedores y atiende al cliente de línea de comandos docker.

4.3 Comandos de Docker

La batería de comandos de Docker es de lo más preguntado de todo el bloque. Conviene dominarla completa:

Comando	Efecto
docker run IMG	Crea e inicia un contenedor desde una imagen, descargándola si no está en local, y ejecuta el comando indicado (= create + start).
docker start ID	Arranca un contenedor YA creado y detenido.
docker stop ID	Detiene un contenedor de forma controlada (SIGTERM; SIGKILL si no responde).
docker kill ID	Detiene INMEDIATAMENTE el contenedor (envía una señal inmediata al proceso de tipo SIGKILL por

	defecto).
docker restart ID	Reinicia un contenedor.
docker rm ID	Elimina un contenedor (debe estar parado, salvo con -f).
docker rmi IMG	Elimina una imagen local.
docker ps	Lista los contenedores EN EJECUCIÓN (moderno: docker container ls).
docker ps -a	Lista TODOS los contenedores, también los detenidos.
docker images	Lista las imágenes descargadas en local.
docker pull IMG	Descarga una imagen desde el registro (Docker Hub).
docker push IMG	Sube una imagen al registro.
docker search término	Busca imágenes en Docker Hub.
docker exec ID cmd	Ejecuta un comando dentro de un contenedor en marcha.
docker logs ID	Muestra los registros del contenedor.
docker build -t IMG .	Construye una imagen a partir de un Dockerfile.

⚠️ **CLAVE DE EXAMEN · 2019 · P13 (SP) · 2026 · P64**

PREGUNTA ORIGINAL *"¿Qué comando de Docker permite crear e iniciar un nuevo contenedor, descargando la imagen si es necesario, y ejecutar en el contenedor un comando especificado?"*

docker run crea e inicia un contenedor a partir de una imagen en un solo paso, descargándola si hace falta, y ejecuta el comando indicado — docker start solo arranca uno ya creado y docker exec ejecuta dentro de uno en marcha. Cayó en 2019 (arrancar un Windows Server desde la imagen SW2008: docker run SW2008) y de nuevo en 2026.

⚠️ **CLAVE DE EXAMEN · 2018 · P15 (SP) · 2019 · P14 (SP)**

Listar contenedores: **docker ps** muestra los que están en ejecución (2018 · P15 (SP)); con **-a** lista todos, también los detenidos (2019 · P14 (SP)).

⚠️ **CLAVE DE EXAMEN · 2018 · P16 (SP) · 2026 · P7 (SP) · 2026 · P8 (SP)**

Detener contenedores: **docker stop ID** lo para de forma controlada — SIGTERM y, si no termina, SIGKILL (2018 · P16 (SP)); **docker kill ID** lo interrumpe **inmediatamente** con SIGKILL (2026 · P7 (SP)). Una vez parado, **docker rm ID** lo elimina (2026 · P8 (SP)).

⚠️ **CLAVE DE EXAMEN · 2018 · P17 (SP) · 2026 · P9 (SP) · 2018 · P33 (SP) (R)**

Imágenes: **docker search jenkins** busca en Docker Hub (2018 · P17 (SP)); **docker pull ImagenA** descarga una imagen al equipo local (2026 · P9 (SP)); **docker images** lista las imágenes locales

(2018 · P33 (SP) (R)).

💡 **TRUCO MNEMOTÉCNICO**

Docker: **run = crear y arrancar; start = solo arrancar**. Para parar: **stop es educado (SIGTERM), kill es fulminante (SIGKILL)**. Y las parejas: ps/ps -a (corriendo/todos), rm/rmi (contenedor/imagen), pull/push (bajar/subir).

5. Orquestación de contenedores: Kubernetes

5.1 Qué es Kubernetes

Cuando hay decenas o miles de contenedores hace falta un **orquestador**. **Kubernetes** (abreviado K8s, de Google, hoy de la CNCF, Cloud Native Computing Foundation) es la plataforma estándar de **gestión de contenedores**: automatiza su despliegue, escalado, balanceo de carga, autorreparación y actualización.

OpenShift es la plataforma de contenedores de **Red Hat construida sobre Kubernetes**, que añade herramientas de desarrollo y seguridad. **Rancher** gestiona clústeres de Kubernetes (crear clústeres, añadir o quitar máquinas, desplegar aplicaciones, desde un único lugar). Google Cloud tiene GKE (Google Kubernetes Engine), Amazon Web Services tiene EKS (Elastic Kubernetes Service) y Microsoft Azure tiene AKS (Azure Kubernetes Service) como servicios de Kubernetes gestionados en la nube.

⚠️ **CLAVE DE EXAMEN · 2017 · P72 · 2017 · P73**

Kubernetes = gestión (orquestación) de contenedores — nada que ver con MIMO ni con LUNs (2017 · P72). La plataforma de gestión de contenedores de Red Hat sobre Kubernetes se denomina **OpenShift** (2017 · P73).

5.2 Arquitectura del clúster

Un clúster de Kubernetes es un conjunto de máquinas (servidores) que trabajan juntas para ejecutar servicios y aplicaciones. En su arquitectura, se distinguen el **plano de control** (los componentes que toman las decisiones globales) de los **nodos de trabajo** (donde corren los contenedores, y los que realmente ejecutan las aplicaciones):

Componente	Función
kube-apiserver	Plano de control: expone la API del clúster; todo pasa por él.
etcd	Plano de control: almacén clave-valor con el estado del clúster.
kube-scheduler	Plano de control: decide EN QUÉ NODO se ejecuta cada Pod según los recursos disponibles y las restricciones.
kube-controller-manager	Plano de control: ejecuta los controladores que corrigen el estado (réplicas, nodos caídos...).
kubelet	En cada nodo: agente que arranca y vigila los

	contenedores del nodo.
kube-proxy	En cada nodo: gestiona las reglas de red de los Servicios.
Container runtime	En cada nodo: motor que ejecuta los contenedores (containerd, CRI-O).

⚠️ **CLAVE DE EXAMEN · 2026 · P65**

El componente del plano de control que **decide en qué nodo se ejecuta un Pod** según los recursos disponibles es el **kube-scheduler** — el apiserver expone la API, etcd almacena el estado y el controller-manager ejecuta los controladores.

5.3 Objetos de Kubernetes y sus manifiestos

Kubernetes se administra de forma **declarativa**: los objetos se describen en manifiestos YAML (aplicados con kubectl apply -f). Los objetos básicos son el **Pod** (la unidad mínima: uno o varios contenedores que comparten red y almacenamiento), el **Deployment** (mantiene el número deseado de réplicas de un Pod) y el **Service** (punto de acceso estable a un conjunto de Pods). Todo manifiesto tiene cuatro campos obligatorios:

- **apiVersion**: versión de la API con la que se define el objeto.
- **kind**: tipo de objeto (Pod, Service, Deployment...).
- **metadata**: los valores que **identifican unívocamente al objeto**: name, namespace, labels.
- **spec**: el estado deseado del objeto.

⚠️ **CLAVE DE EXAMEN · 2022 · P65**

El campo del objeto Kubernetes que contiene los valores que **identifican unívocamente al objeto** (name, namespace) es **metadata** — kind es el tipo, apiVersion la versión de la API y spec el estado deseado.

5.4 Protocolos de los Servicios

Los **Services** de Kubernetes exponen los Pods usando protocolos de **transporte**: **TCP (por defecto)**, **UDP** y **SCTP**.

⚠️ **CLAVE DE EXAMEN · 2024 · P13 (SP)**

Protocolos utilizables en los servicios de Kubernetes: **SCTP, TCP (por defecto) y UDP**. Las opciones con HTTP/HTTPS/FTP/SSH mezclan protocolos de aplicación, y el «por defecto» siempre es TCP.

💡 **TRUCO MNEMOTÉCNICO**

Kubernetes: el **scheduler** es el que «agenda» cada Pod en su nodo; **etcd** es la libreta donde se apunta todo; **metadata** es el DNI del objeto. Y sus Services hablan **T-U-S**: TCP (por defecto), UDP y SCTP.

6. Microservicios

6.1 Del monolito a los microservicios

Una arquitectura de **microservicios** descompone la aplicación en servicios pequeños e independientes, cada uno con una única responsabilidad, que se comunican por red (API REST, mensajería). Frente al **monolito** (toda la aplicación en un solo bloque), cada microservicio se desarrolla, despliega y **escala por separado** — por eso los contenedores y Kubernetes son su hábitat natural. A cambio, aumenta la complejidad operativa: la observabilidad, la resiliencia y la comunicación entre servicios pasan a ser críticas.

6.2 Composición: orquestación y coreografía

Las funcionalidades se implementan **componiendo** varios microservicios. Existen **dos estrategias** para gestionar esa composición:

- **Orquestación**: un servicio central (el orquestador) **dirige y coordina** las llamadas al resto de microservicios, como un director de orquesta.
- **Coreografía**: **no hay coordinador central**; cada microservicio reacciona a los **eventos** que publican los demás, como bailarines que se sincronizan entre sí.

> ⚠️ **CLAVE DE EXAMEN · 2019 · P63**
> Las dos estrategias de composición de microservicios son **coreografía y orquestación** — «coordinación» y «gestión» son distractores. Orquestación = coordinador central; coreografía = reacción a eventos sin coordinador.

6.3 Piezas habituales del ecosistema

Alrededor de los microservicios aparecen componentes que conviene conocer de nombre: la **API Gateway** (punto de entrada único que enruta las peticiones), el **service discovery** (registro dinámico de dónde está cada servicio), y los patrones de resiliencia como el **circuit breaker** (cortar llamadas a un servicio caído para no propagar el fallo).

Cifras clave	
25 / 465 / 587	SMTP en claro / SMTPS / envío autenticado con STARTTLS (submission).
110 / 995	POP3 en claro / POP3S seguro.
143 / 993	IMAP en claro / IMAPS seguro (TCP).
209	Puerto IANA de QMTP (Quick Mail Transfer Protocol).
250	Código SMTP de acción completada (fin de transmisión OK).
220 / 354	Códigos SMTP: servicio listo / inicio del contenido (DATA).
TCP	Protocolo por defecto de los Services de Kubernetes (también UDP y SCTP).