13. Estrategias de Migración a SaaS

Si bien el atractivo de llegar a SaaS es bien comprendido, determinar cuál es la mejor forma de hacer ese movimiento puede ser más desafiante. Cuando se tiene una oferta existente con clientes e ingresos, este cambio viene acompañado de preocupaciones naturales.

Implicará encontrar un equilibrio entre lo viejo y lo nuevo, y trazar un camino que abra nuevos horizontes sin disrumpir completamente el negocio.

El equilibrio en la migración

Si bien dedicaremos la mayor parte de la energía a los matices técnicos de la migración, la estrategia de migración que se elija debe estar directamente moldeada por una evaluación de los parámetros de negocio, mercado y operacionales que motivan el movimiento hacia SaaS.

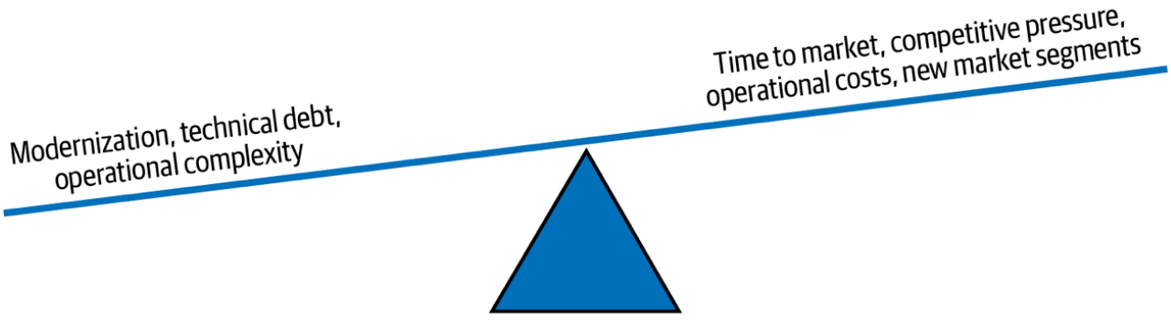

Figure 13-1. The migration balancing act

En última instancia, una buena estrategia de migración requiere una combinación de compromisos técnicos y de negocio.

De hecho, el proceso de seleccionar la estrategia de migración puede representar el primer paso en la migración cultural SaaS de una organización, reuniendo a los equipos de producto, operaciones y tecnología para jugar un rol más colaborativo en la definición del modelo de migración general.

Consideraciones de tiempo

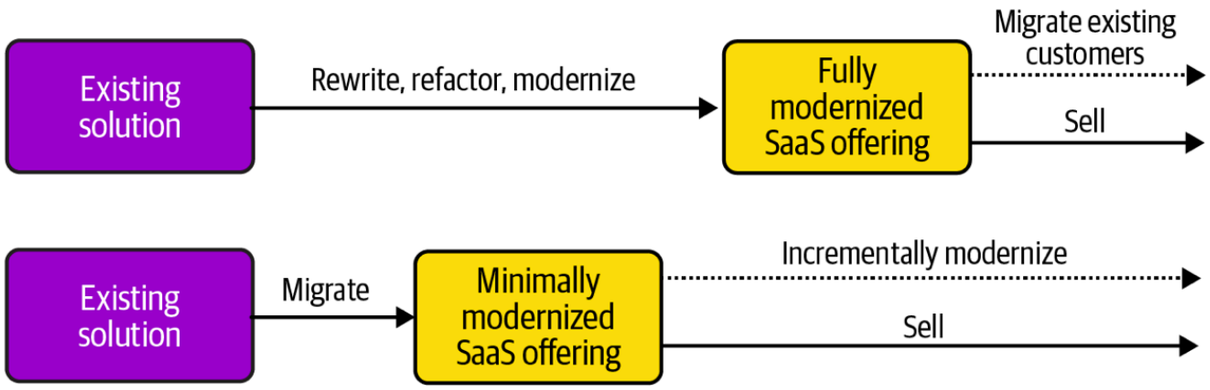

Figure 13-2. Migration timing trade-offs

Con el modelo superior, se moderniza sin la carga adicional de introducir estos cambios sobre una versión ya publicada del producto. Esto brinda mayor libertad, movilidad y menos necesidad de considerar cuidadosamente cómo cada paso hacia la modernización puede incorporarse a una versión en producción de la solución. El segundo modelo lleva a SaaS antes, pero lo hace a expensas de hacer más lenta y compleja la modernización incremental futura.

Lo que se pierde al evaluar estos compromisos, sin embargo, es el valor de la retroalimentación de los clientes. **Con el modelo superior, habrá un período prolongado en el que no se recibirá retroalimentación de los clientes.** Se asume que las decisiones tomadas durante ese tiempo satisfarán sus necesidades y que las necesidades del mercado no evolucionarán significativamente en esa ventana. Esto puede funcionar, pero sin duda conlleva riesgos reales que pueden pasarse fácilmente por alto.

Cuando los equipos eligen hacer el movimiento a SaaS, a menudo están eligiendo cambiar fundamentalmente la forma en que abordan su trabajo. Soporte, operaciones, ventas, product management —todos estos roles pueden, de alguna manera, ser reformulados en función del movimiento hacia un modelo de entrega SaaS.

Como regla general, **prefiero estrategias de migración que pongan mayor énfasis en operar como un negocio SaaS lo antes posible.** La combinación de recibir retroalimentación antes y transformar todo el negocio antes parece ofrecer más valor que enfocarse primero en la modernización.

¿Qué tipo de pez eres?

El movimiento a SaaS puede tener un impacto profundo en los fundamentos del negocio, provocando cambios en dinámicas económicas clave.

La inversión inicial en la transformación a SaaS, el movimiento de clientes hacia nuevas estrategias de monetización y las economías de escala de SaaS pueden influir en la trayectoria financiera de un negocio SaaS.

The Technology-as-a-Service Playbook (Baker & Taylor), de Thomas Lah y J.B. Wood, describe esta dinámica con un diagrama que pone estos aspectos económicos de la migración en mejor perspectiva.

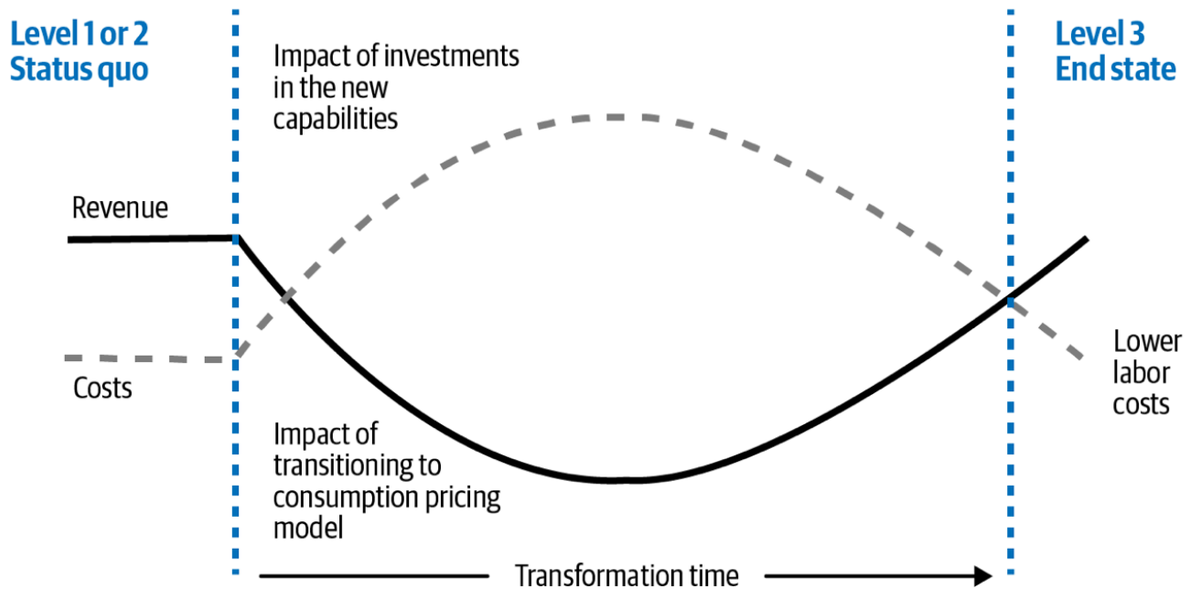

Figure 13-3. The migration fish model

Este diagrama, por su forma, se denomina el modelo del pez. Yuxtapone ingresos y costos, mostrando cómo pueden evolucionar a lo largo de la transformación hacia un modelo de entrega SaaS.

Esto es naturalmente intimidante para muchas empresas en proceso de migración. En general, quieren que ese pez sea lo más delgado posible. Las buenas noticias aparecen en el lado derecho del diagrama. La expectativa es que, a medida que se construyen todas las eficiencias y mecanismos que crearán un entorno multi-tenant altamente ágil, los costos eventualmente tenderán a la baja.

La clave, sin embargo, es utilizar este modelo para preguntarse qué tipo de pez se quiere ser. Esto ayuda a enmarcar la discusión general de migración e influye notablemente en la mentalidad general del esfuerzo de transformación.

Más allá de la transformación tecnológica

He dejado claro que la migración a SaaS consiste en gran medida en pasar de una experiencia centrada en el producto a una mentalidad más orientada al negocio y al servicio.

- Los equipos de ventas y marketing, por ejemplo, definitivamente necesitarán considerar cómo cambiarán los fundamentos de comercializar y vender un producto con un modelo SaaS.
- El soporte al cliente suele experimentar una transformación, adoptando más un modelo de Customer Success que desplaza la mentalidad desde la resolución de problemas del cliente hacia el compromiso proactivo con los clientes y la gestión de su experiencia.
- El pricing y la facturación también formarán parte del proceso de transformación, requiriendo considerar cómo SaaS influirá en el modelo de monetización general del negocio.

Patrones de migración

Es importante señalar que ninguno de los patrones aboga por una migración big bang en la que se abandone todo y se reescriba la mayor parte de la solución. **Eso no sería coherente con una mentalidad de migración. Tampoco es una estrategia que yo recomendaría.**

El movimiento a SaaS tiene muchos aspectos en movimiento, y adoptar un enfoque más incremental permite a los equipos evolucionar su estrategia en función de los datos que obtienen al ver partes de su solución cobrar vida en un entorno de trabajo.

El enfoque incremental también permite ver cómo los elementos operacionales y de aplicación del entorno emergen en paralelo, lo que facilita evaluar si las estrategias están abordando las necesidades funcionales y operacionales del servicio.

Los cimientos

Cada migración —independientemente de la estrategia elegida— incluye la introducción de un control plane.

Esto es especialmente importante en un escenario de migración, ya que el control plane representa un área completamente nueva e independiente que el negocio debe construir como parte del movimiento hacia una arquitectura multi-tenant.

Este control plane es lo que permite introducir la noción fundamental de tenancy en el entorno, proporcionando los elementos que permiten gestionar y operar los tenants desde un panel único de control.

Por lo tanto, el primer paso en el proceso es determinar qué capacidades del control plane serán necesarias para soportar los requisitos iniciales de la oferta SaaS.

Con una mentalidad de migración, probablemente esto comenzará como un conjunto más reducido de servicios que brindan soporte base para la multi-tenancy, sin construir toda la profundidad que eventualmente se querrá en el control plane. Luego, el control plane puede evolucionar de forma incremental a medida que la solución madura.

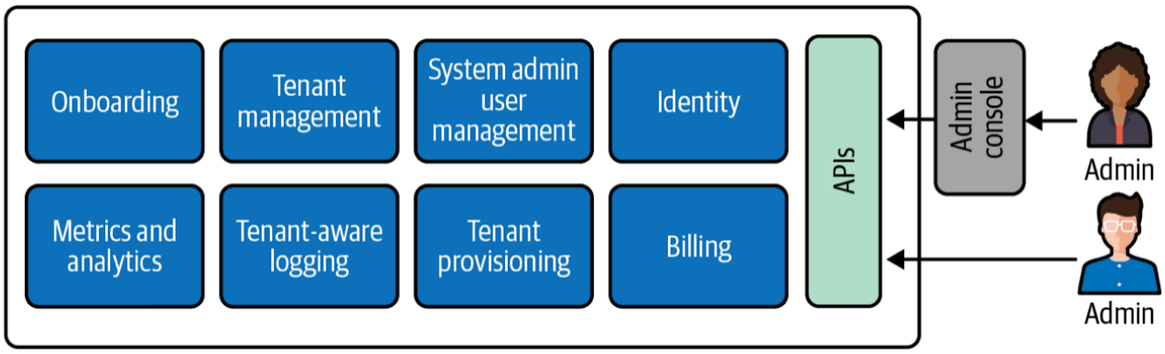

Figure 13-4. Initial control plane services

Los servicios enfocados en orquestar el onboarding y la creación de tenants son puntos focales importantes, ya que soportan el aprovisionamiento de recursos de tenant, la creación de tenants y el establecimiento de los elementos centrales del modelo de identidad SaaS.

Esto también sienta las bases para la autenticación de tenants y la inyección del contexto de tenant en las solicitudes de la aplicación.

La migración del modelo de identidad puede requerir algo más de reflexión. Es probable que la solución legacy que se está migrando ya incluya soporte para alguna forma de autenticación. La clave, sin embargo, es que incluso si se mantiene el modelo de identidad actual, habrá que determinar cómo se ampliará la experiencia de autenticación para soportar el contexto de tenant.

Otros servicios (métricas y analíticas, logging con conciencia de tenant y facturación) pueden ser clave para las necesidades a largo plazo del entorno multi-tenant. **Sin embargo, se pueden implementar versiones simplificadas de estos para establecer el concepto y luego mejorarlos a medida que el sistema y las necesidades evolucionen.**

Finalmente, en el lado derecho del diagrama se ve una experiencia de administración. Esto está pensado como un marcador de posición para la experiencia de gestión y configuración que se expondrá para el control plane. Es fundamental tener algunos aspectos de esto en funcionamiento al inicio del camino de migración. Esto forzará a ejercitar la experiencia de autenticación del administrador del sistema y permitirá exponer perspectivas multi-tenant clave que serán de ayuda durante la migración.

Lift-and-shift en modo silo

Este patrón, como su nombre sugiere, está enfocado exclusivamente en mover una carga de trabajo existente hacia un modelo de entrega SaaS con el mínimo de refactorización posible.

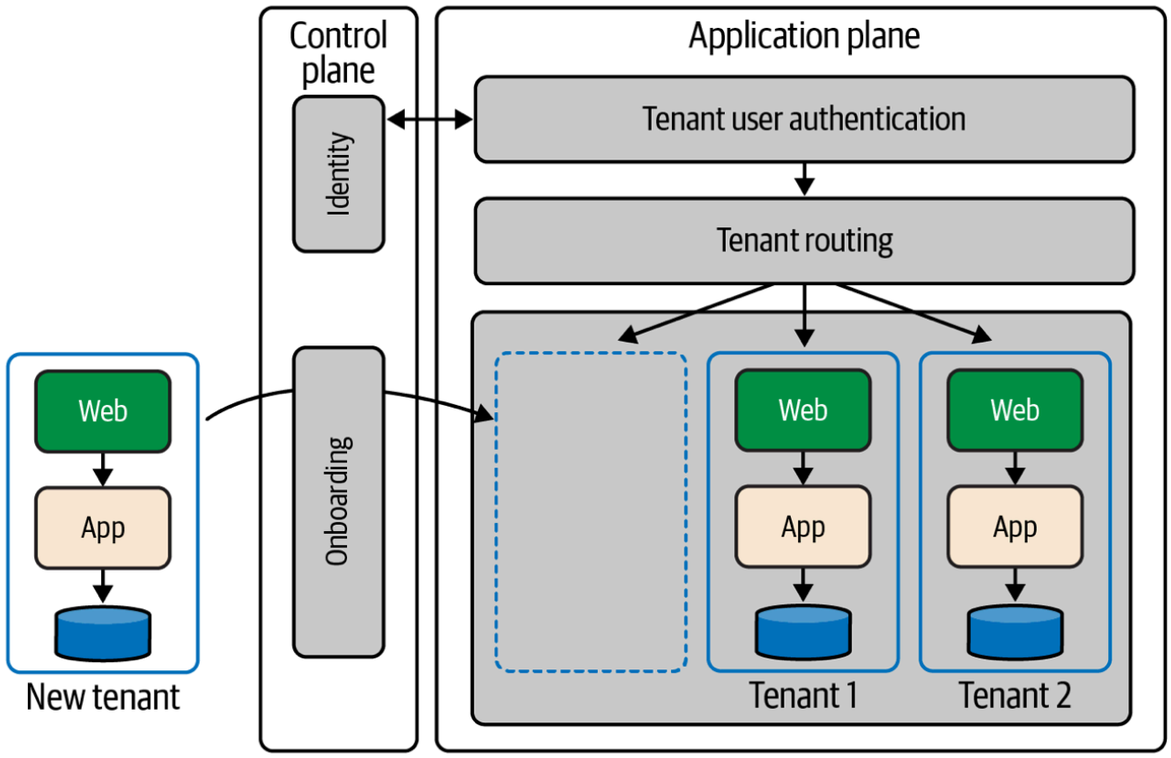

Figure 13-5. Silo lift-and-shift migration

La clave es que cada stack de tenant sea creado por un proceso de onboarding completamente automatizado que esencialmente aprovisioné, configure y despliegue estos silos de tenant en el application plane.

Una advertencia importante que puede ser menos obvia es que cada tenant en este patrón de lift-and-shift en modo silo está ejecutando la misma versión de la aplicación. Esto, de hecho, puede ser un punto de tensión significativo en el paso de un modelo legacy instalado por tenant a un entorno completamente multi-tenant.

La buena noticia de este modelo es que a menudo representa el camino más rápido y menos invasivo hacia SaaS para algunas organizaciones.

Si bien este puede ser un movimiento de bajo impacto, es importante señalar que el entorno actual requerirá cambios para participar en este modelo multi-tenant.

Este contexto debe agregarse al logging y a otras perspectivas operacionales para ayudar a los equipos a centralizar las perspectivas de tenant y proporcionar un enfoque universal para la resolución de problemas con contexto de tenant.

Migración por capas

La diferencia clave aquí es que, como parte de la migración, se elige profundizar un poco más en el código para introducir elementos puntuales de optimización que permitan obtener algunas eficiencias de costo y operacionales durante la migración.

Algunos patrones de arquitectura se adaptan mejor al modelo por capas que otros. La Figura 13-6 proporciona un ejemplo de cómo se podría aplicar la migración por capas en una arquitectura n-tier clásica.

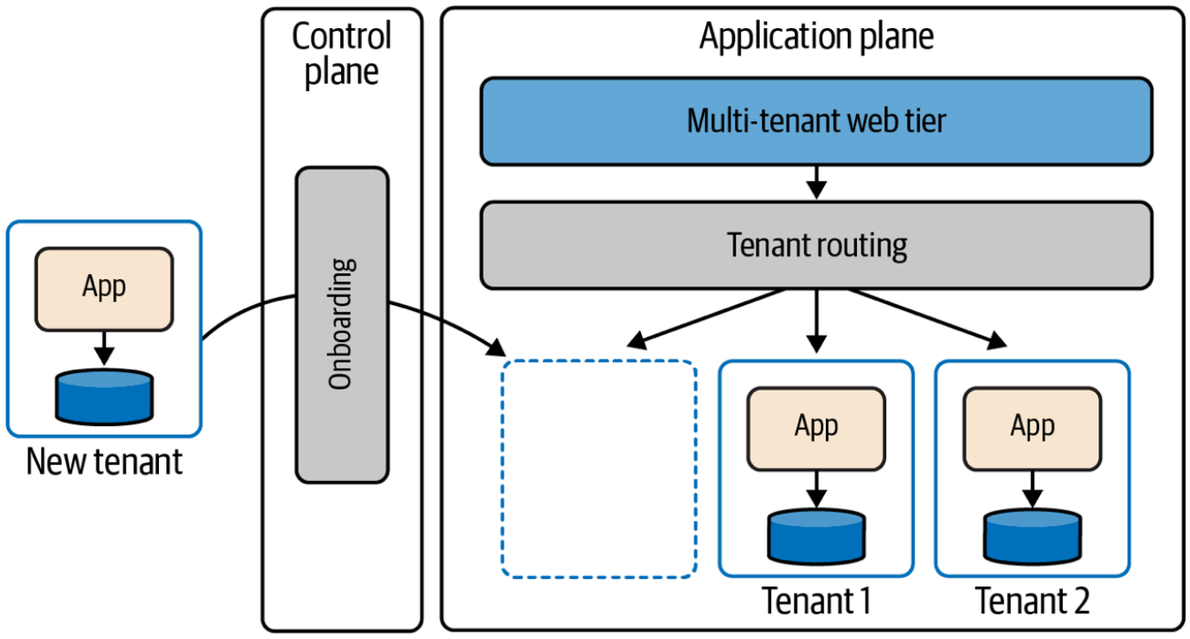

Figure 13-6. A layered migration of the web tier

La diferencia es que algunas capas del stack de la aplicación se han movido a silos. Sin embargo, también se puede observar que la capa web de la arquitectura n-tier no está incluida en el silo ni como parte del onboarding. En cambio, con este enfoque por capas, **he migrado la capa web hacia un modelo multi-tenant compartido (pooled) (mostrado en la parte superior del diagrama).**

Lo que hizo posible esto (en este ejemplo) fue que el movimiento hacia un modelo pooled podía lograrse sin necesidad de una refactorización significativa. Si se requieren cambios más significativos, la capa puede no representar un buen candidato para la migración.

Ahora, imaginemos continuar por este camino por capas, migrando más capas del stack hacia un modelo compartido.

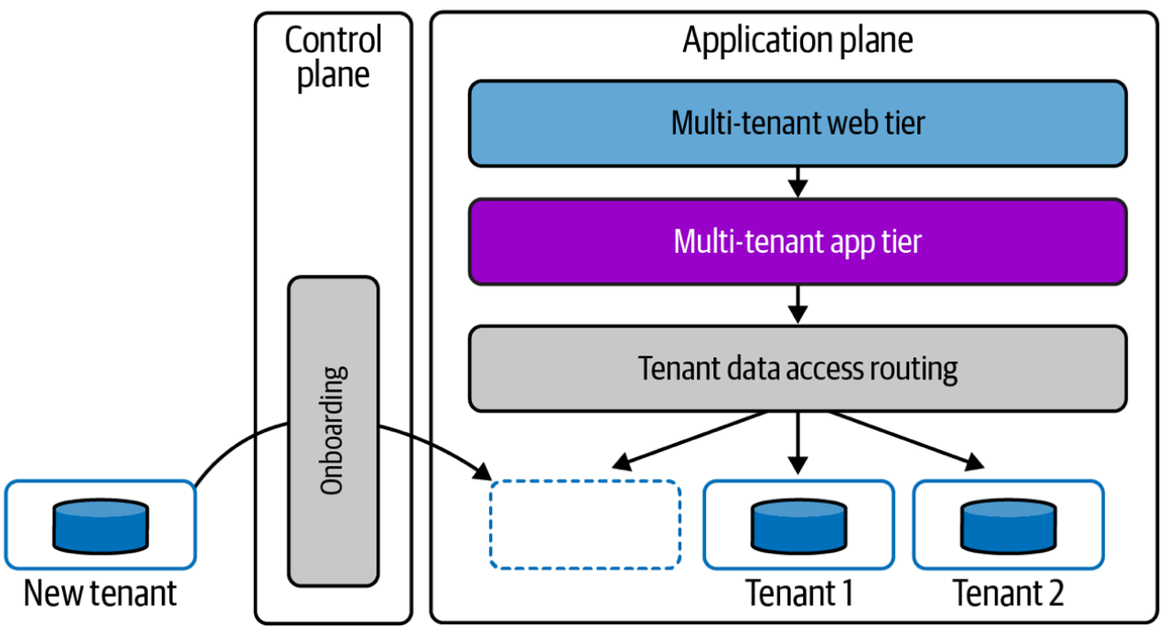

Figure 13-7. Migrating the application layer

Suponiendo que esto encajara con el entorno, el onboarding ahora solo requeriría aprovisionar nuevo almacenamiento para cada tenant. La capa de aplicación ahora usaría el contexto de tenant entrante de cada solicitud para conectar la llamada de la aplicación con los recursos de almacenamiento de tenant correspondientes.

Migración servicio a servicio

Algunos equipos tienen un mayor enfoque en la migración hacia una arquitectura modernizada. Estos equipos buscan un camino más directo e inmediato hacia una arquitectura que maximice la eficiencia en costos, la agilidad, la innovación y el perfil operacional que representa un entorno multi-tenant de vanguardia.

Los equipos pueden considerar que tienen más tiempo, o pueden enfrentar desafíos con su diseño actual que exigen una reescritura más inmediata.

Este patrón adopta el enfoque de modernizar incrementalmente los servicios individuales de la arquitectura, ejecutando el código existente junto con los nuevos microservicios modernizados.

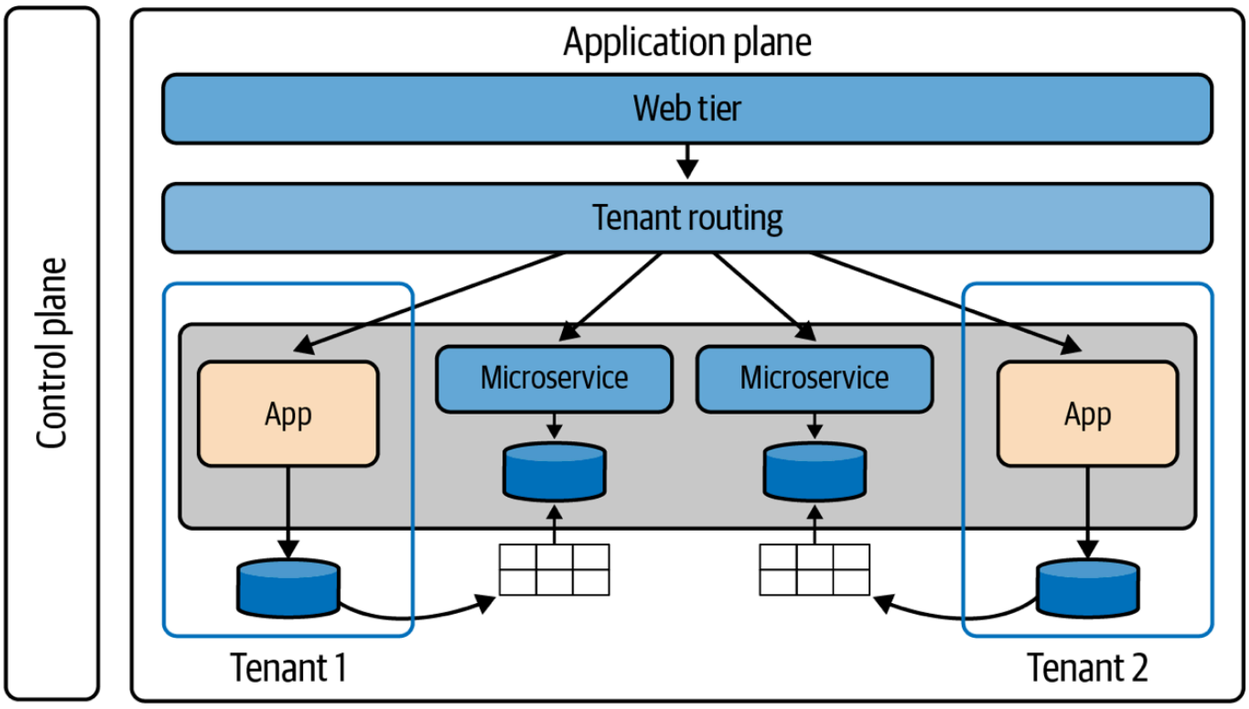

Figure 13-8. Service-by-service migration

Aquí, en lugar de mover la capa de aplicación a una capa compartida única, comenzamos por modernizar la funcionalidad que reside en esa capa de aplicación, extrayendo incrementalmente funcionalidad de estas capas de aplicación siloed y moviéndola hacia microservicios multi-tenant pooled.

Se puede elegir un área del sistema con el menor potencial de disrumpir el negocio, o un área que represente un cuello de botella clave que requiera atención más inmediata. No hay absolutos aquí.

Es fácil imaginar cómo el código de la capa de aplicación ha evolucionado a lo largo de los años y cuán estrechamente acopladas pueden estar las diferentes partes de la funcionalidad. **Sin límites bien definidos, los desarrolladores SaaS pueden haber aprovechado el acceso abierto a cualquier parte de la capa de aplicación.**

Para algunos, esto significará hacer compromisos en torno a estos primeros servicios para comenzar a descomponer la capa de aplicación. Eso puede requerir comenzar con servicios más granulares de lo deseado.

A medida que se vayan extrayendo más partes de la capa de aplicación, esto se irá facilitando. Luego, una vez que estos servicios estén en funcionamiento, se pueden usar cargas de trabajo reales para encontrar los mejores lugares donde descomponer aún más el sistema.

Una parte importante de la creación de estos servicios también está enfocada en extraer los datos de los silos de tenant, creando nuevos modelos de almacenamiento multi-tenant. Con los nuevos microservicios, los datos se moverán hacia almacenamiento pooled o siloed multi-tenant según las necesidades de cada servicio. **Esto le da al microservicio la autonomía que necesita y nos permite gestionar todo el acceso a los datos a través del contrato del servicio.**

Además de la extracción de microservicios, también hay que considerar cómo la migración influirá en la estrategia de enrutamiento de tenant de la aplicación.

Una vez que se tienen los fundamentos de este modelo en funcionamiento, gran parte del enfoque estará en poner en marcha el siguiente conjunto de microservicios. La parte positiva de este enfoque es que pone al equipo en el camino hacia la modernización y lo obliga a comenzar a abordar las realidades multi-tenant antes que después.

Sin compromisos en los nuevos microservicios

Los nuevos microservicios que se introducen durante la migración están pensados para ser construidos como servicios completamente modernizados. Estos servicios no deben hacer compromisos que puedan socavar los objetivos de modernización.

La guía que ofrezco es construir estos microservicios como si se estuvieran creando para un entorno greenfield. Los diseñaría, implementaría y desplegaría usando las estrategias y patrones de mejores prácticas que habilitarán los objetivos de eficiencia, escalabilidad y agilidad a largo plazo del entorno. **Habrá algunas áreas donde puede ser necesario soportar pequeñas variaciones temporales para funcionar junto con el entorno legacy, pero no hay que dejar que esos compromisos se filtren demasiado en el diseño.**

Esto es especialmente importante para los primeros microservicios. Estos servicios iniciales son el modelo para la siguiente oleada de servicios que se crearán, por lo que conviene exigirles mucho. **Deben establecer las bibliotecas base que puedan ser usadas por otros microservicios.**

Integración del código legacy con el control plane

En este modelo de integración servicio a servicio, habrá código nuevo y viejo ejecutándose en paralelo. El código nuevo tendrá soporte multi-tenant completo. Estos nuevos servicios dependerán en gran medida de las capacidades transversales del control plane.

Las partes legacy de la capa de aplicación no tienen conciencia del control plane y, tal como están construidas, no tienen ninguna noción de tenancy. Esto podría significar que, durante la migración, el equipo necesitaría construir vistas operacionales a partir de estos dos elementos diferentes de la arquitectura. **Esto impondría una carga adicional significativa a los equipos de operaciones y haría generalmente difícil monitorear el estado del sistema.**

Esto puede llevar a los equipos a invertir en la creación de nuevas integraciones con el control plane en el código de la capa de aplicación existente, sabiendo que ese código eventualmente será reemplazado.

Comparación de patrones

Cada uno de los patrones de migración descritos tiene fortalezas y debilidades claras. He tratado de destacar los factores de negocio y técnicos que acompañan a cada uno de estos patrones.

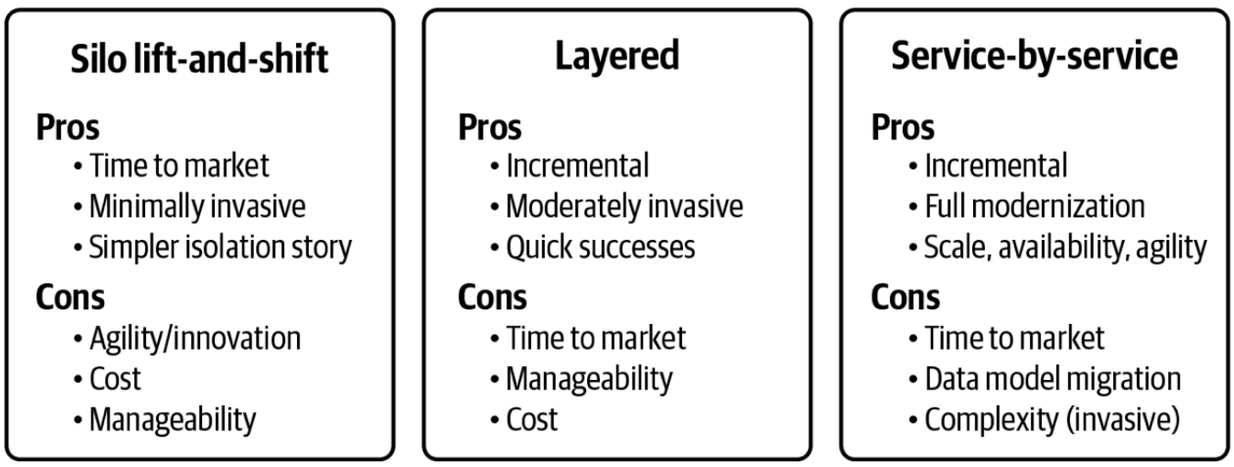

Figure 13-11. Migration pattern pros and cons

El punto de partida importa

Una vez elegido el camino y la estrategia, aún existen opciones sobre cómo secuenciar los pasos de la migración. **Mi preferencia es hacia una estrategia de migración que comience con un enfoque claro en inyectar tenancy y establecer los elementos de la experiencia operacional general.**

Para mí, esto siempre comienza con la creación del control plane. Eso no significa construir cada elemento del control plane. Se trata más de incorporar soporte para los elementos base que pondrán el sistema en funcionamiento en un modelo multi-tenant desde el primer día. Esto coloca la multi-tenancy en primer plano desde el día uno y tiene un efecto en cascada sobre todos los pasos restantes del proceso de migración.

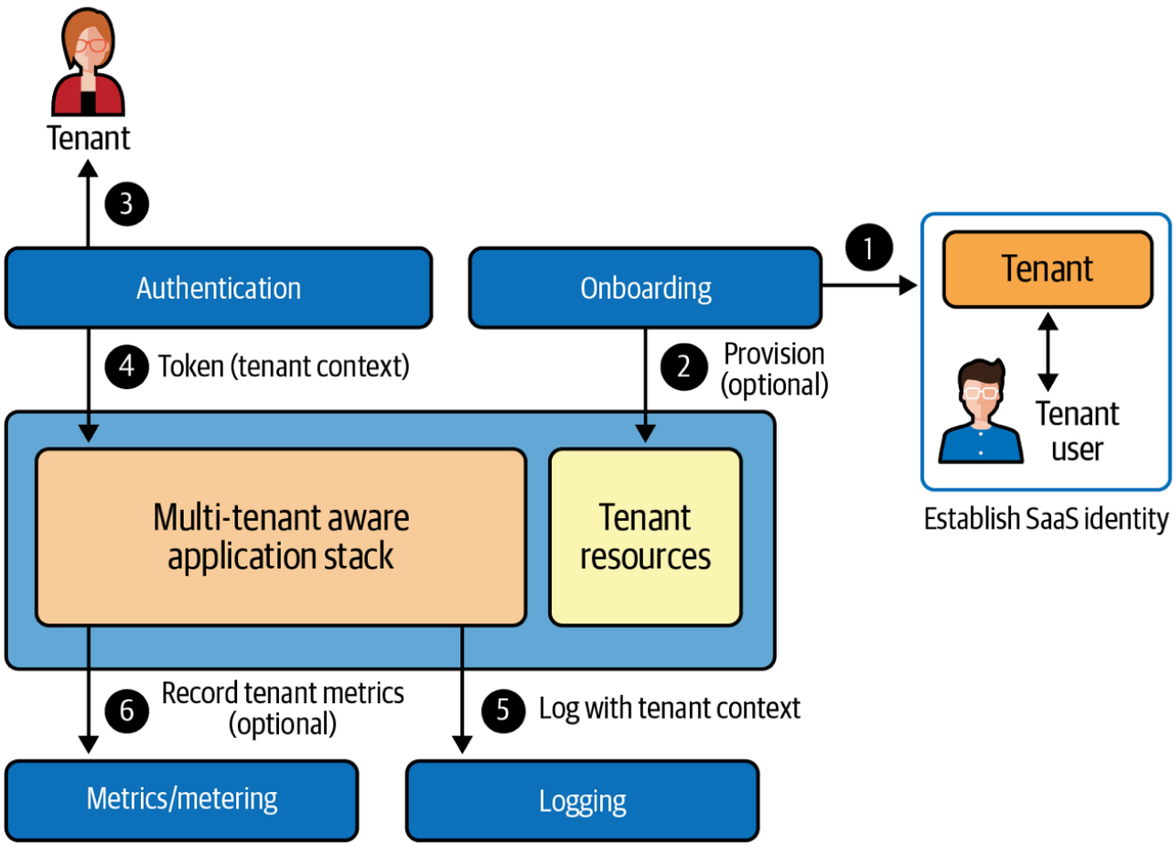

Figure 13-13. SaaS building blocks

1. Primero, en la parte superior derecha, se ve la idea de establecer una identidad SaaS. El nuevo entorno debe comenzar soportando una forma clara para que los tenants se incorporen a la solución, creen un tenant, creen un usuario de tenant y vinculen ese usuario al tenant (paso 1).
2. El otro elemento de onboarding mostrado está conectado con la idea de aprovisionar y configurar infraestructura a medida que cada nuevo tenant se incorpora (paso 2). **He presentado esto como opcional ya que algunos entornos pueden no requerir aprovisionamiento de recursos por tenant.**
3. En la parte superior izquierda del diagrama se ve la autenticación de tenant (paso 3). Este es también un elemento base que debe abordarse al inicio de la construcción de la migración.
4. Esto se conecta con el trabajo de onboarding discutido, autenticando a un usuario y obteniendo su identidad SaaS, que luego puede pasarse al application plane. Tener esto en funcionamiento temprano obliga a los equipos a considerar cómo procesarán y aplicarán el contexto de tenant a medida que fluye hacia los servicios backend y la arquitectura del application plane (paso 4).
5. Aquí se muestra un ejemplo simple de aplicación del contexto de tenant en los servicios de la aplicación. Primero, vemos una ilustración de logs que se publican con contexto de tenant (paso 5). Esto es clave para garantizar que el entorno, desde el principio, incluya soporte para exponer la actividad del sistema con contexto de tenant.
6. El otro ejemplo destaca el uso del contexto de tenant para publicar datos de métricas y analíticas (paso 6). Estos son los datos que proporcionan perspectivas de negocio y operacionales sobre la salud y la actividad de los tenants.

La mera presencia de estos mecanismos generará preguntas que deben salir a la superficie temprano en la migración de la solución. Esto evita que los equipos vean la tenancy, el logging de tenants, las métricas y otros conceptos clave multi-tenant como algo que se añade después de que la aplicación está en funcionamiento.