3. Modelos de despliegue multi-tenant

Seleccionar tu modelo de despliegue multi-tenant es una de las primeras cosas que harás como arquitecto de SaaS.

¿Qué es un modelo de despliegue?

Parte del desafío de describir arquitecturas SaaS es que no hay una única estrategia de arquitectura que de alguna manera se aplique a todas las soluciones SaaS.

Es tu trabajo determinar qué permutaciones de estas estrategias se ajustan mejor a las necesidades de tu solución.

El primer obstáculo que enfrenté en este espacio fue la ausencia de cualquier terminología precisa que categorizara con precisión los diferentes patrones de despliegues multi-tenant.

Un modelo de despliegue indica cómo estarás desplegando recursos e infraestructura dentro del application plane de tu solución multi-tenant.

La conclusión clave aquí es que tu modelo de despliegue representa la estrategia que estás usando para determinar cómo las cargas de trabajo de los tenants serán mapeadas a sus recursos de infraestructura correspondientes. Expresa qué recursos serán compartidos y cuáles serán dedicados.

Las decisiones que tomes al seleccionar un modelo de despliegue tendrán una influencia profunda en casi todas las dimensiones de nuestra oferta SaaS.

Selección de un modelo de despliegue

Sin importar dónde comiences tu camino hacia SaaS, ciertamente hay algunos factores globales más amplios que influenciarán tu selección de modelo de despliegue. Los objetivos de empaquetamiento, tiering y precios, por ejemplo, a menudo juegan un rol clave en determinar qué modelo(s) de despliegue podrían ajustarse mejor a tus objetivos de negocio. El costo y la eficiencia operacional también son parte del rompecabezas del modelo de despliegue.

La clave aquí es que no debes estar buscando un modelo de despliegue único para tu aplicación. En su lugar, debes comenzar con las necesidades de tu dominio, clientes y negocio y trabajar hacia atrás hasta la combinación de requisitos que te señale hacia el modelo de despliegue que se ajuste a tus objetivos actuales y aspiracionales.

También es importante notar que se espera que el modelo de despliegue de tu entorno SaaS evolucione con el tiempo. Preocúpate menos por hacerlo bien desde el día uno y simplemente espera que estarás usando datos de tu entorno para encontrar oportunidades de refactorizar tu modelo de despliegue.

Introducción a los modelos Silo y Pool

Verás que usaré el término "silo" para referirme a cualquier modelo donde un recurso está dedicado a un tenant dado. Usaré el término "pool" para referenciar cualquier modelo donde un recurso de tenant es compartido por uno o más tenants.

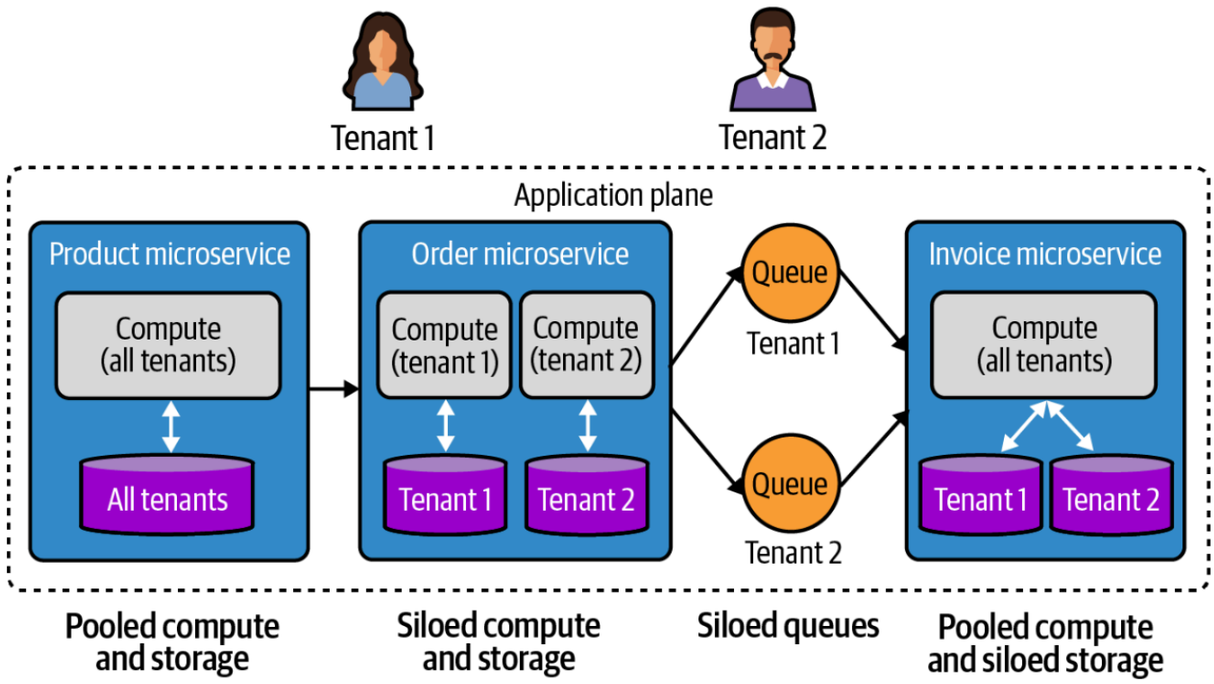


Figure 3-2. Siloed and pooled resource models

La conclusión clave es que los términos "silo" y "pool" se usan para caracterizar generalmente la huella de arquitectura de uno o más recursos. Estos términos pueden ser aplicados de una manera muy granular, destacando cómo la tenancy se mapea a elementos muy específicos de tu arquitectura.

Despliegue Full Stack Silo

Como su nombre sugiere, el modelo full stack silo coloca a cada tenant en un entorno donde todos sus recursos están completamente aislados en silos.

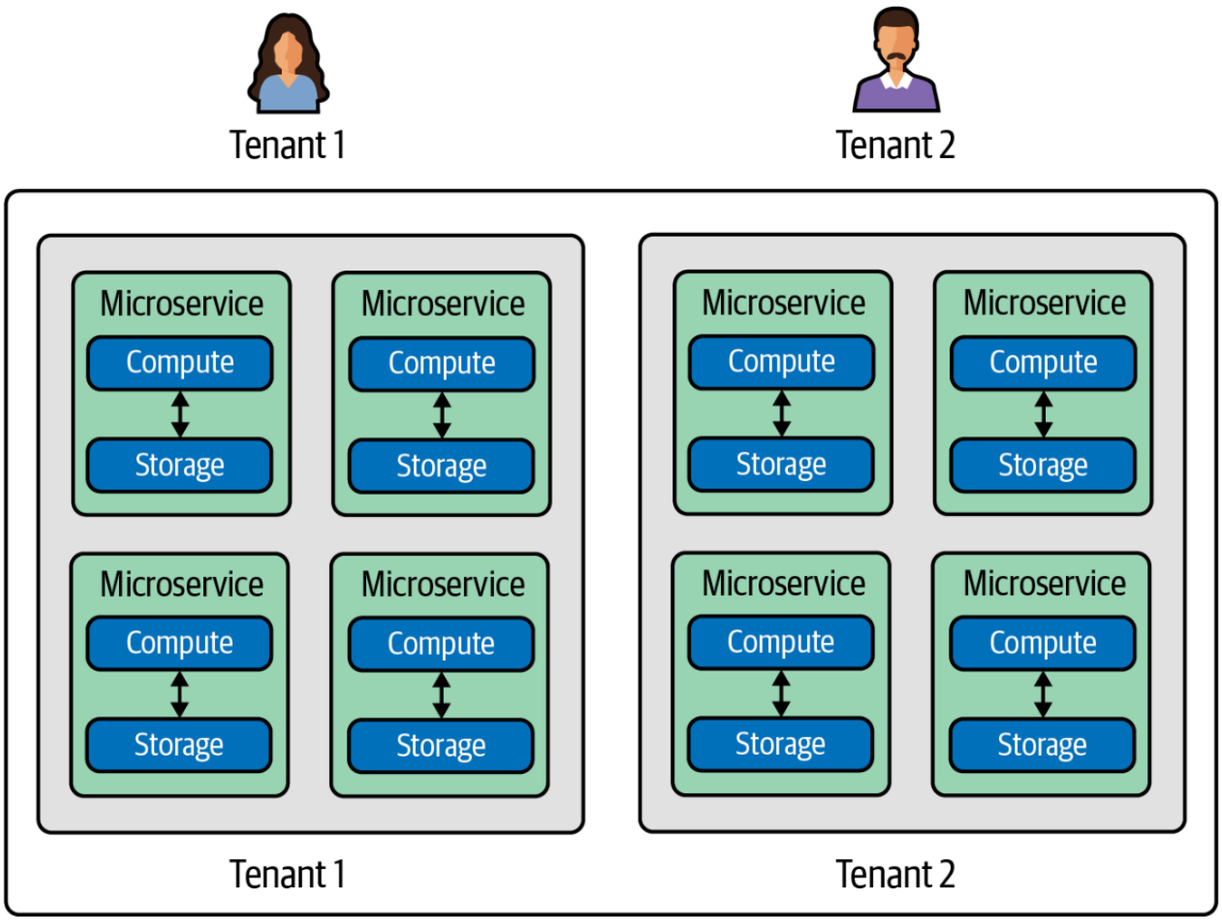


Figure 3-3. Full stack silo deployment model

En realidad, lo que está en el silo podría ser representado por cualquier número de tecnologías y estrategias de diseño diferentes. Esto podría haber sido un entorno de n-capas con capas de web, aplicación y almacenamiento separadas. Podría haber incluido cualquier número de diferentes modelos de cómputo y otros servicios también (colas, object storage, mensajería, etc.).

Dónde encaja Full Stack Silo

El modelo full stack silo puede sentirse un poco como un antipatrón de SaaS. Después de todo, gran parte de nuestra discusión sobre SaaS se centra en la agilidad y eficiencia.

- Las consideraciones de cumplimiento (compliance) y legacy son dos de las razones típicas por las que los equipos terminarán optando por una huella full stack silo.
- El modelo full stack silo también puede representar un buen ajuste para organizaciones que están migrando una solución legacy a SaaS.
- Full stack silo también puede ser una estrategia de tiering. Por ejemplo, algunas organizaciones pueden ofrecer una versión premium de su solución que, por el precio correcto, ofrecerá a los tenants una experiencia completamente dedicada.

Consideraciones de Full Stack Silo

Complejidad del control plane

Ahora, con el modelo full stack silo, debes considerar cómo la naturaleza distribuida del modelo full stack influenciará la complejidad de tu control plane.

Dado que nuestra solución se está ejecutando en un modelo de silo por tenant, el application plane debe soportar entornos completamente separados para cada tenant. En lugar de interactuar con un único recurso compartido, nuestro control plane debe tener cierta conciencia de cada uno de estos silos de tenant.

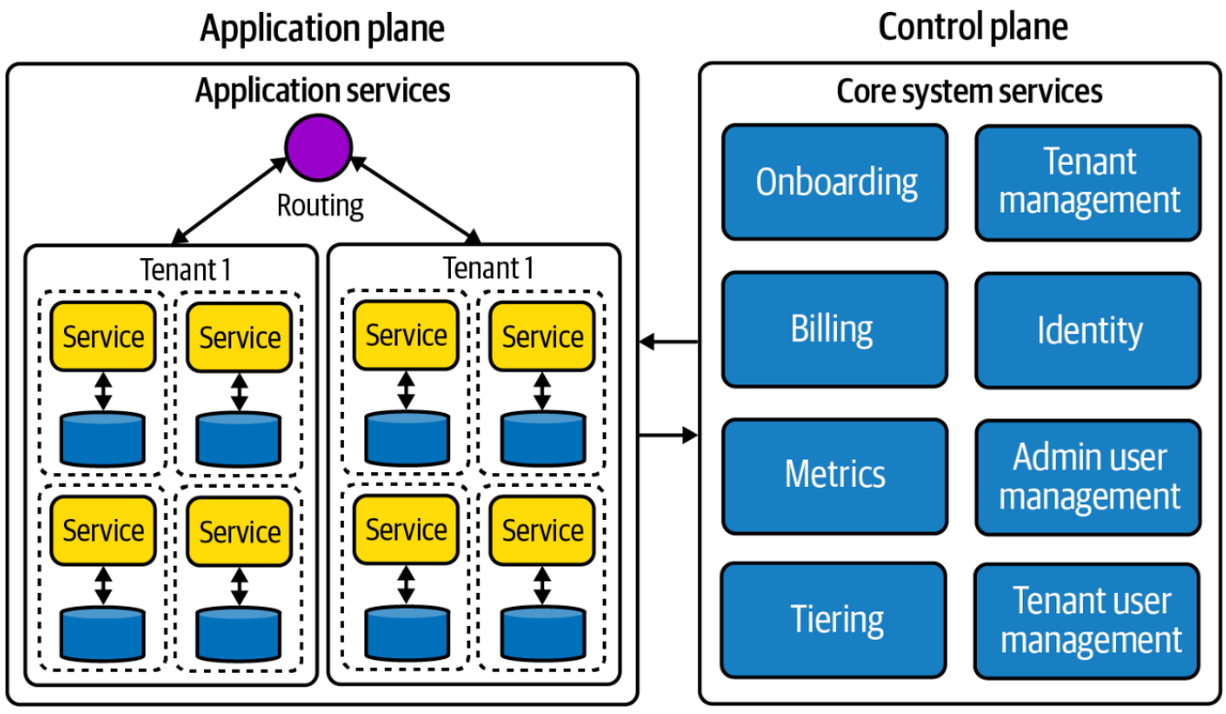


Figure 3-4. Managing and operating a full stack silo

Imagina implementar el onboarding de tenant en este ejemplo.

- La adición de cada nuevo tenant a este entorno debe aprovisionar y configurar completamente cada entorno de tenant aislado en silo.
- Esto también complica cualquier herramienta que pueda necesitar monitorear y gestionar la salud de los entornos de tenant. Tu código del control plane debe saber dónde se puede encontrar cada entorno de tenant y estar autorizado para acceder a cada silo de tenant.
- Cualquier herramienta operacional que tengas que se use para gestionar recursos de infraestructura también necesitará lidiar con la huella más grande y más distribuida de estos entornos de tenant, lo que podría hacer que la resolución de problemas y gestión de recursos de infraestructura sea difícil de manejar.
- Cualquier herramienta que hayas creado para centralizar métricas, logs y analíticas para tus tenants también necesitará poder agregar los datos de estos entornos de tenant separados.
- El despliegue de actualizaciones de aplicación también es más complicado en este modelo. Tu código de DevOps tendrá que implementar actualizaciones en cada silo de tenant.

Impactos en el escalado

Siempre que estés aprovisionando infraestructura separada para cada tenant, necesitas pensar en cómo este modelo escalará a medida que agregues más tenants.

Si estás ejecutando Kubernetes, por ejemplo, podría reducirse a cuán efectivamente los constructos de silo de Kubernetes escalarían aquí (clusters, namespaces, etc.). Si estás usando redes de nube separadas o constructos de cuenta para cada tenant aislado en silo, tendrás que considerar cualquier límite y restricción que pueda ser aplicada por tu proveedor de nube.

Consideraciones de costo

Aunque hay medidas que puedes tomar para limitar el sobreaprovisionamiento de entornos aislados en silo, este modelo sí pone límites en tu capacidad de maximizar las economías de escala de tu entorno SaaS.

Para cada tenant, entonces, tendrás un conjunto de costos base para tenants que incurrirás, incluso si no hay carga en el sistema.

Además, como estos entornos no son compartidos, no obtenemos las eficiencias que vendrían con distribuir la carga de muchos tenants a través de infraestructura compartida que escala basándose en la carga de todos los tenants.

Como parte de las fórmulas de costo, también debemos considerar cómo el modelo full stack silo impacta la eficiencia operacional de tu organización.

Consideraciones de routing

Con un full stack silo, necesitarás considerar cómo el tráfico será enrutado a cada silo basándose en el tenant context. Aunque hay cualquier número de diferentes constructos de red que podemos usar para enrutar esta carga, aún necesitarás considerar cómo esto será configurado.

- ¿Estás usando subdominios para cada tenant?
- ¿Tendrás un dominio compartido con el tenant context embebido en cada solicitud?

La configuración de este constructo de routing debe ser completamente dinámica. A medida que cada nuevo tenant es incorporado a tu sistema, necesitarás actualizar la configuración de routing para soportar el enrutamiento de este nuevo tenant a

su silo correspondiente.

Disponibilidad y radio de impacto

El modelo full stack silo sí ofrece algunas ventajas cuando se trata de la disponibilidad general y durabilidad de tu solución.

La naturaleza dedicada del modelo silo te da la oportunidad de contener algunos problemas a entornos de tenant individuales. Esto ciertamente puede tener un efecto positivo general en los perfiles de disponibilidad de tu servicio.

El despliegue de nuevos lanzamientos también se comporta un poco diferente en entornos aislados en silo. En lugar de tener tu lanzamiento empujado a todos los clientes al mismo tiempo, el modelo full stack silo puede liberar a los clientes en oleadas.

Tener que desplegar a cada silo por separado requiere que tengas un proceso de despliegue más complicado que puede, en algunos casos, socavar la disponibilidad de tu solución.

Atribución de costos más simple

Un beneficio significativo del modelo full stack silo es su capacidad de atribuir costos a tenants individuales.

Saber cuánto de una base de datos compartida o recurso de cómputo fue consumido por un tenant dado no es tan fácil de inferir en entornos pool.

Los proveedores de nube y herramientas de terceros son generalmente buenos mapeando costos a recursos de infraestructura individuales y calculando un costo para cada tenant.

Full Stack Silo en acción

El modelo de cuenta por tenant

En este modelo, las cuentas a menudo se ven como los límites más estrictos que se pueden crear entre tenants. Esto, para algunos, hace que una cuenta sea un hogar natural para cada tenant en tu modelo full stack silo.

La cuenta permite que cada silo de tus entornos de tenant esté rodeado y protegido por todos los mecanismos de aislamiento que los proveedores de nube usan para aislar las cuentas de sus clientes.

Esto limita el esfuerzo y la energía que tendrás que gastar para implementar el aislamiento de tenant en tu entorno SaaS.

Atribuir costos de infraestructura a tenants individuales también se convierte en un proceso mucho más simple en un modelo de cuenta por tenant.

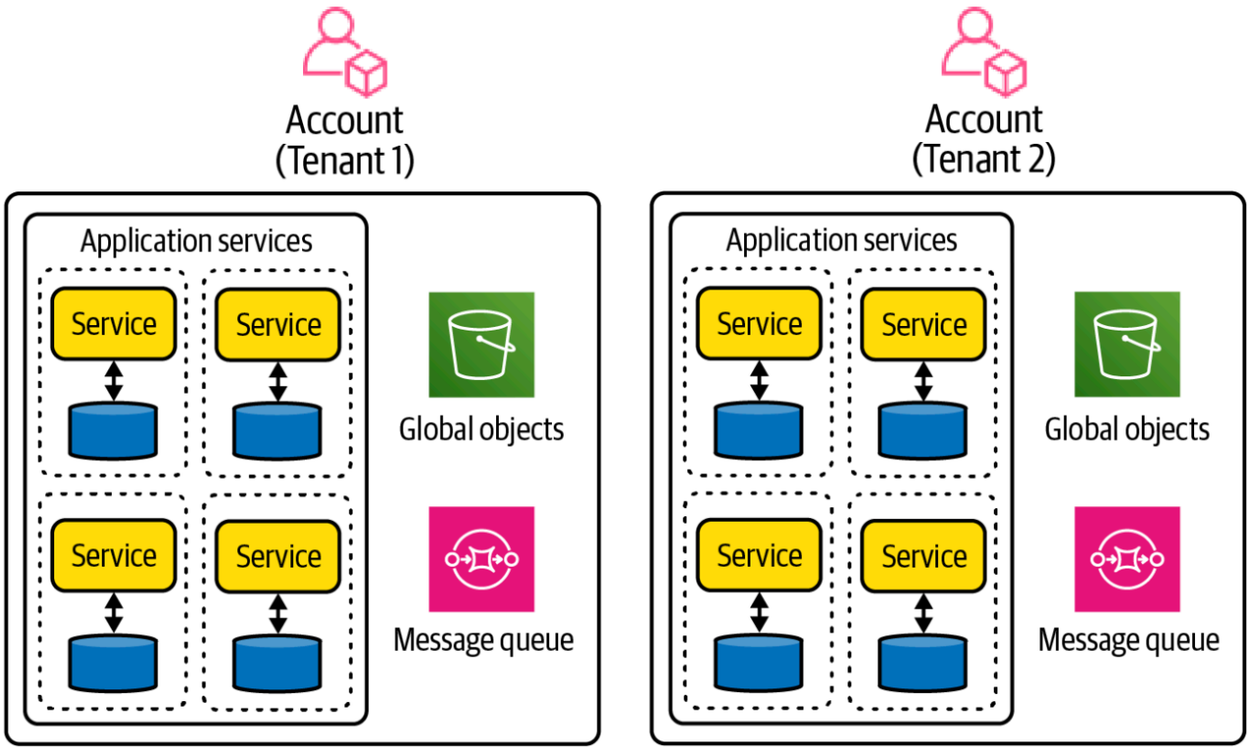


Figure 3-5. The account-per-tenant full stack silo model

En algunos casos, puedes encontrar razones específicas para tener algunas partes de tu infraestructura ejecutándose en algún modelo compartido. Sin embargo, cualquier cosa que sea compartida no puede tener un impacto en los requisitos de rendimiento, cumplimiento y aislamiento de nuestra experiencia full stack silo.

Las decisiones que tomes aquí deben comenzar evaluando la intención de tu modelo full stack silo.

- ¿Elegiste este modelo basándote en una expectativa de que los clientes querrían que toda su infraestructura estuviera completamente separada de otros tenants,
- o fue más basado en un deseo de evitar problemas de vecino ruidoso (noisy neighbor) y requisitos de aislamiento de datos?

Tus respuestas a estas preguntas tendrán una influencia significativa en cómo elijas compartir partes de tu infraestructura en este modelo.

En términos generales, mapear cuentas a tenants podría verse como un antipatrón. Las cuentas, para muchos proveedores de nube, no fueron necesariamente diseñadas para ser usadas como el hogar de tenants en entornos SaaS multi-tenant.

El problema más básico que puedes enfrentar aquí es que podrías exceder el número máximo de cuentas soportado por tu proveedor de nube.

La proliferación de cuentas puede terminar socavando la agilidad y eficiencia de tu negocio SaaS. Imagina tener cientos o miles de tenants ejecutándose en este modelo. **Esto se traducirá en una huella masiva de infraestructura que necesitarás gestionar.**

El modelo de Virtual Private Cloud (VPC) por tenant

El modelo que veremos ahora, el modelo de Virtual Private Cloud (VPC) por tenant, es uno que depende más de constructos de redes para albergar la infraestructura que pertenece a cada uno de nuestros tenants aislados en silo.

Dentro de la mayoría de entornos de nube se te da acceso a una rica colección de constructos de redes virtualizadas que pueden ser usados para construir, controlar y asegurar la huella de tus entornos de aplicación. **Estos constructos de redes proporcionan mecanismos naturales para implementar una implementación full stack en silo.**

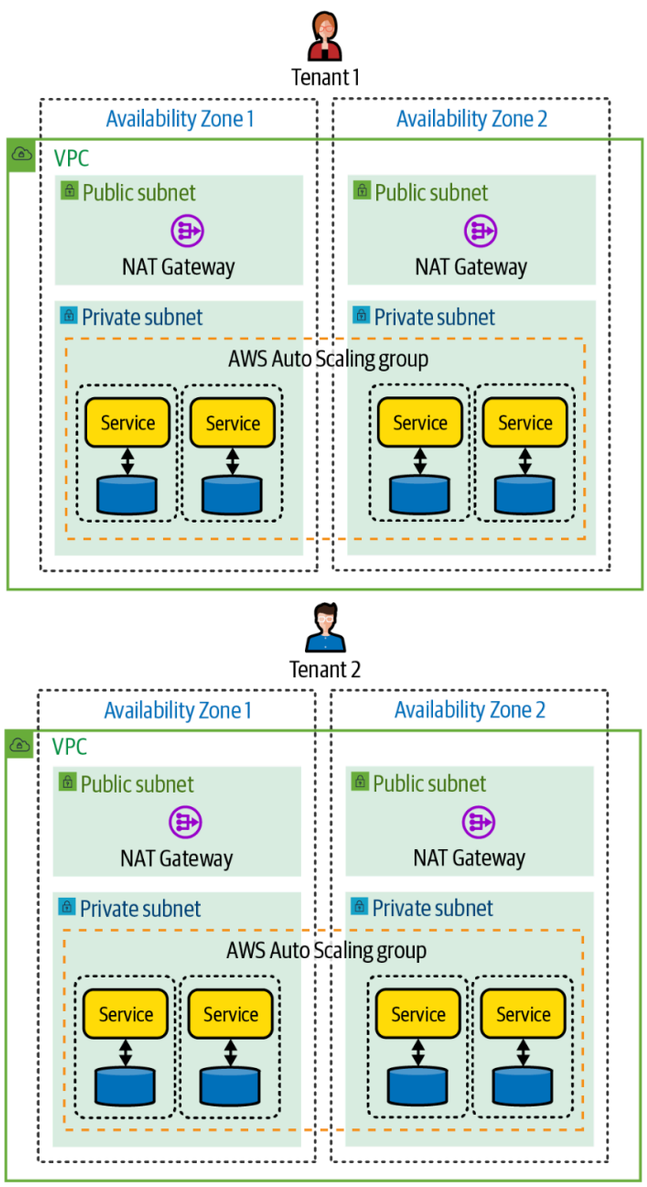


Figure 3-6. The VPC-per-tenant full stack silo model

También quería ilustrar la huella de alta disponibilidad de nuestro VPC al hacer que incluya dos Zonas de Disponibilidad (AZs) separadas.

También tenemos subnets separadas para separar las subnets públicas y privadas de nuestra solución.

He incluido todos estos detalles de red para resaltar **la idea de que estamos ejecutando nuestros tenants en silos de red que ofrecen a cada uno de ellos un entorno de redes aislado y resiliente** que se apoya en toda la bondad de redes virtualizadas que viene con construir y desplegar tu solución en un modelo de silo de VPC por tenant.

Aunque este modelo puede parecer menos rígido que el modelo de cuenta por tenant, en realidad te proporciona un conjunto sólido de constructos para prevenir cualquier acceso entre tenants. Como parte de su propia naturaleza, estas herramientas de redes te permiten crear ingress y egress muy cuidadosamente controlados para tus entornos de tenant.

Otro modelo que aparece ocasionalmente es el modelo de subnet por tenant. Aunque rara vez veo este modelo, hay algunas instancias donde los equipos pondrán cada silo de tenant en una subnet dada.

Automatización de onboarding

En el modelo de VPC por tenant, el enfoque está más en aprovisionar tus constructos de VPC y desplegar tus servicios de aplicación. Eso probablemente seguirá siendo un proceso pesado, pero la mayoría de lo que necesitas crear y configurar puede lograrse a través de un proceso completamente automatizado.

Consideraciones de escalado

Así como hay límites en el número de cuentas que puedes tener, puede haber límites en el número de VPCs que puedes tener.

Tener infraestructura de tenant expandiéndose a través de cientos de VPCs puede impactar la agilidad y eficiencia de tu experiencia SaaS.

Permaneciendo alineados con una mentalidad de Full Stack Silo

El full stack silo solo existe para acomodar realidades de dominio, cumplimiento (compliance), tiering, y cualquier otra realidad de negocio que pueda justificar el uso de un modelo full stack silo.

En todos los aspectos, un entorno full stack silo es tratado de la misma manera que un entorno pool.

- Siempre que se lancen nuevas funcionalidades, se despliegan a todos los clientes.
- Si tu configuración de infraestructura necesita ser cambiada, ese cambio debe aplicarse a todos tus entornos aislados en silo.
- Si tienes políticas para escalado u otros comportamientos de runtime, se aplican basándose en los tiers de tenant.

Nunca deberías tener una política que se aplique a un tenant individual.

El modelo Full Stack Pool

Con el modelo full stack pool, ahora veremos entornos SaaS donde **todos los recursos del application plane para nuestros tenants están ejecutándose en un modelo de infraestructura compartida.**

Es aquí donde el enfoque está directamente en lograr economías de escala, eficiencias operacionales, beneficios de costo, y un perfil de gestión más simple que son los subproductos naturales de un modelo de infraestructura compartida.

Sin embargo, con nuestro full stack pool, este contexto es esencial para cada operación que se realiza. Acceder a datos, registrar mensajes de log, grabar métricas. Todas estas operaciones necesitarán resolver el contexto de tenant actual en runtime para completar exitosamente su tarea.

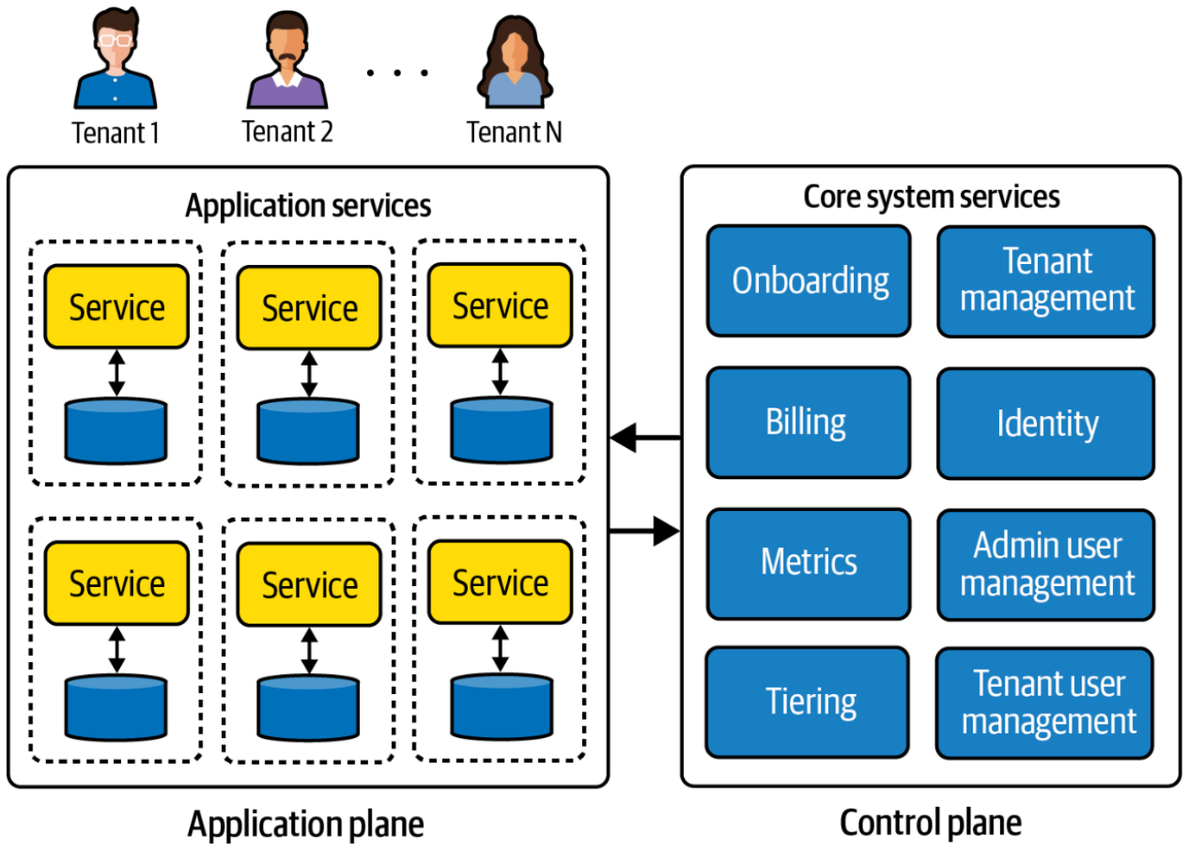


Figure 3-7. A full stack pool model

La idea fundamental es que cuando tenemos un recurso pool, ese recurso pertenece a múltiples tenants. **Como resultado, el contexto de tenant es necesario para aplicar alcance y contexto a cada operación en runtime.**

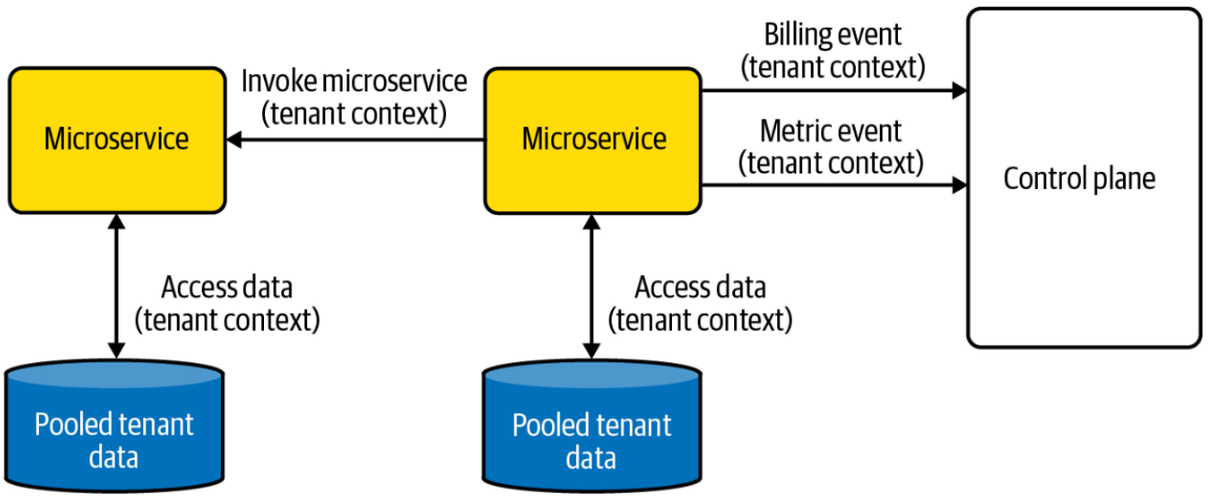


Figure 3-8. Tenant context in the full stack pooled environment

Consideraciones del Full Stack Pool

Escala

En un escenario ideal, tu sistema tendría, en un momento dado en el tiempo, solo suficientes recursos asignados para acomodar la carga actual que están imponiendo los tenants.

Habría cero recursos sobreaprovisionados, lo que permitiría al negocio optimizar márgenes y asegurar que la adición de nuevos tenants no generaría un pico en los costos que pudiera socavar la rentabilidad del negocio.

Esto no es práctico ni realista, pero es la mentalidad que a menudo rodea el modelo full stack pool.

Aunque hay ventajas significativas de escalado a obtener en un modelo full stack pool, el esfuerzo para hacer que este escalado sea una realidad puede ser desafiante de realizar completamente. Necesitarás trabajar duro para crear una estrategia de escalado que pueda optimizar la utilización de recursos sin impactar la experiencia del tenant.

Aislamiento

Es importante notar que como parte de adoptar un modelo full stack pool, te enfrentarás a un rango de nuevas consideraciones de aislamiento que pueden influenciar tu diseño y arquitectura.

La suposición es que las economías de escala y eficiencias del modelo pool compensan cualquier sobrecarga adicional y complejidad asociada con aislar recursos pool.

Disponibilidad y radio de impacto

Cualquier interrupción o problema que aparezca en un entorno full stack pool es probable que impacte a todos tus clientes y podría potencialmente dañar la reputación de tu negocio SaaS.

A medida que consideras adoptar un modelo full stack pool, necesitas entender que te estás comprometiendo a un estándar más alto de DevOps, pruebas y disponibilidad que hace todo lo posible para asegurar que tu sistema pueda prevenir, detectar y recuperarse rápidamente de cualquier interrupción potencial.

El riesgo e impacto de cualquier interrupción en un entorno full stack pool demanda un mayor enfoque en asegurar que tu equipo pueda entregar una experiencia de cero tiempo de inactividad. **Esto incluye adoptar estrategias de integración continua/despliegue continuo (CI/CD) de primera clase que te permitan lanzar y revertir nuevas funcionalidades de manera regular sin impactar la estabilidad de tu solución.**

Generalmente, verás a equipos de full stack pool inclinándose hacia estrategias tolerantes a fallos que permiten a sus microservicios y componentes limitar el radio de impacto de problemas localizados. **Aquí, verás mayor aplicación de interacciones asíncronas entre servicios, estrategias de respaldo (fallback), y patrones bulkhead siendo usados para localizar y gestionar interrupciones potenciales de microservicios**.

Vecino ruidoso (Noisy neighbor)

Los entornos full stack pool dependen de políticas de escalado cuidadosamente orquestadas que aseguren que tu sistema agregará y removerá capacidad efectivamente basándose en la actividad de consumo de tus tenants.

Las necesidades cambiantes de los tenants junto con el potencial influjo de nuevos tenants significa que las políticas de escalado que tienes hoy pueden no aplicar mañana.

Cada sistema multi-tenant debe emplear estrategias que les permitan anticipar picos y abordar lo que se conoce como condiciones de vecino ruidoso (noisy neighbor).

Cuando todo es compartido, el potencial para condiciones de vecino ruidoso es mucho mayor. **Debes ser especialmente cuidadoso con el dimensionamiento y perfil de escalado de tus recursos ya que todo debe ser capaz de reaccionar exitosamente a cambios en la actividad de consumo de tenant.**

Esto significa contabilizar y usar tácticas defensivas para asegurar que un tenant no esté saturando tu sistema e impactando la experiencia de otros tenants.

Atribución de costos

Asociar y rastrear costos a nivel de tenant es una propuesta mucho más desafiante en un entorno full stack pool. **Aunque muchos entornos te dan herramientas para mapear tenants a recursos de infraestructura específicos, típicamente no soportan mecanismos que te permitan atribuir consumo a los tenants individuales que están consumiendo un recurso compartido.**

La eficiencia de un modelo full stack pool también viene con nuevos desafíos alrededor de entender la huella de costo de tenants individuales.

Simplificación operacional

He hablado sobre esta necesidad de un panel único (single pane of glass) que proporcione una vista operacional y de gestión unificada de tu entorno multi-tenant. Construir esta experiencia operacional requiere que los equipos ingesten métricas, logs y otros datos que pueden ser presentados en esta experiencia centralizada.

Crear estas experiencias operacionales en un entorno full stack pool tiende a ser una experiencia más simple.

El despliegue también es más simple en el entorno full stack pool. Lanzar una nueva versión de un microservicio simplemente significa desplegar una instancia de ese servicio al entorno pool.

Una arquitectura de ejemplo

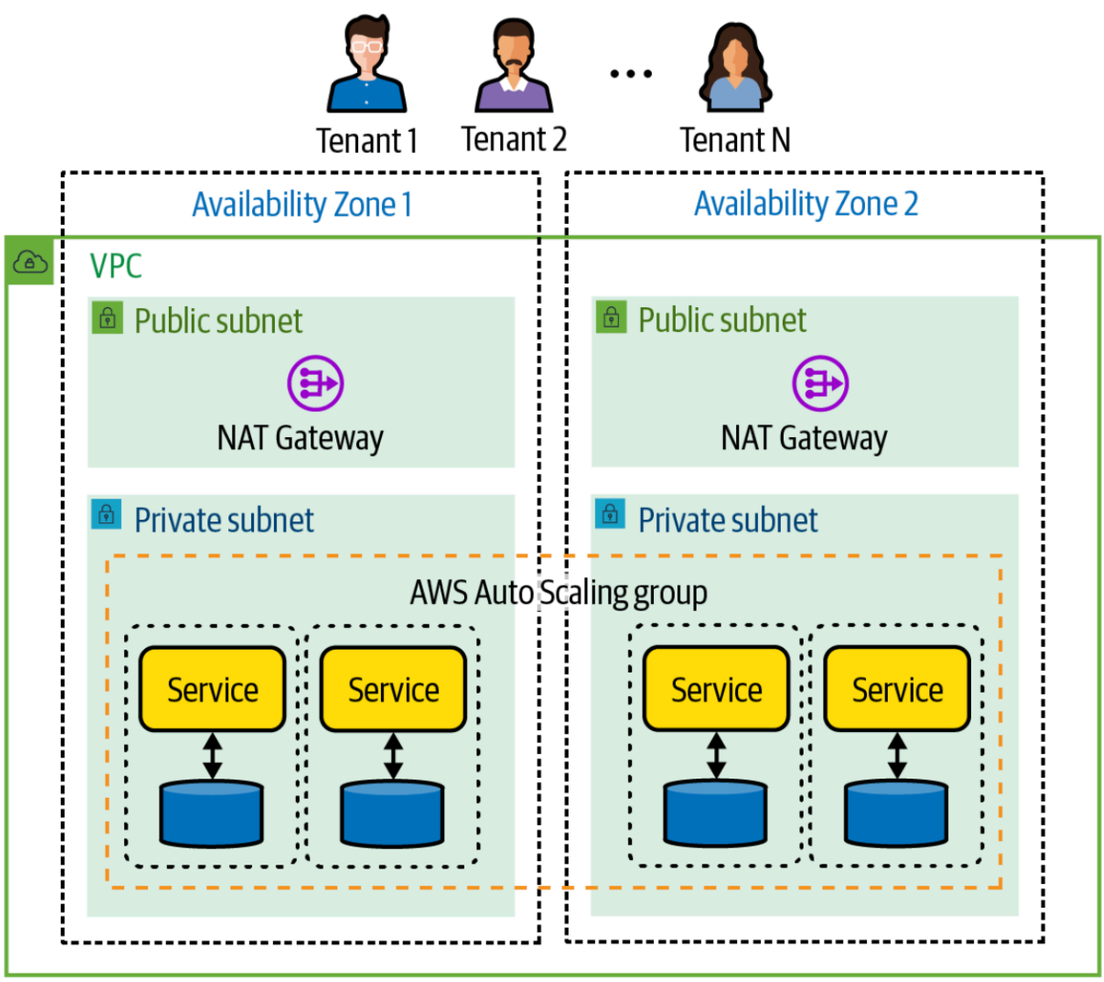


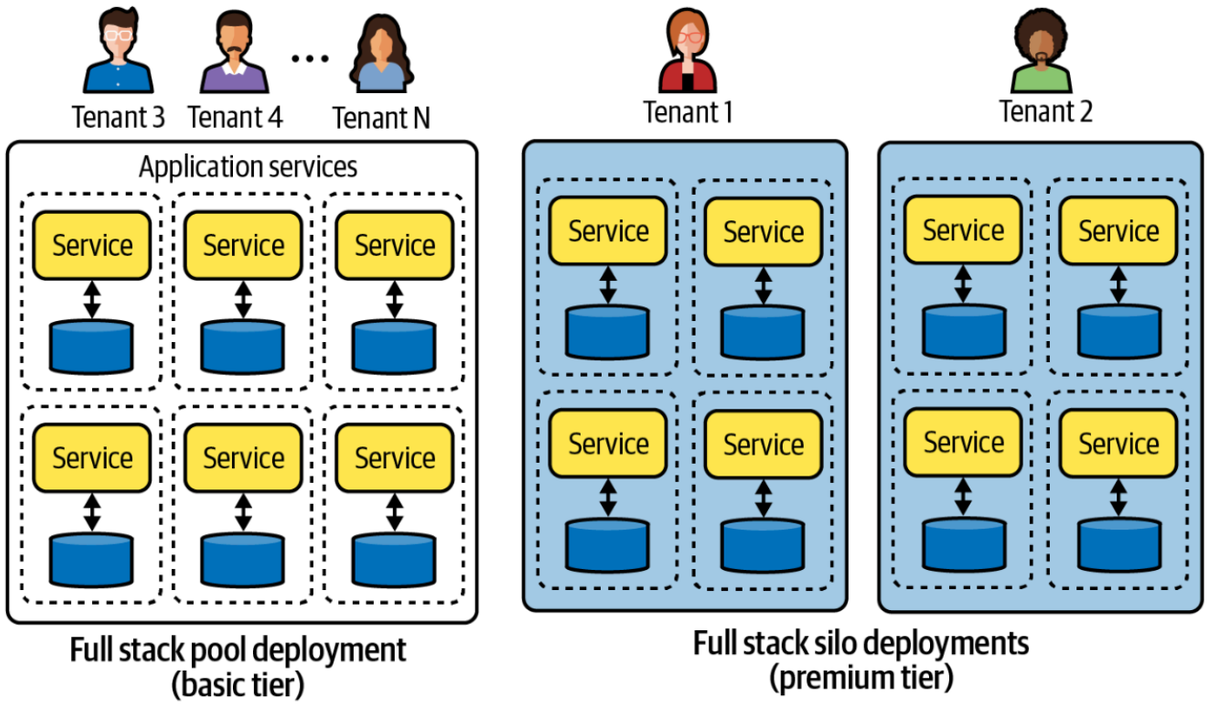
Figure 3-9. A full stack pool architecture

Esta arquitectura representa solo un sabor de un modelo full stack pool. Cada stack tecnológico (contenedores, serverless, almacenamiento relacional, almacenamiento NoSQL, colas) puede influenciar la huella del entorno full stack pool.

El espíritu del full stack pool permanece igual a través de la mayoría de estas experiencias. Ya sea que estés en un cluster de Kubernetes o en un VPC, los recursos en ese entorno estarán en pool y necesitarán escalar basándose en la carga colectiva de todos los tenants.

Un modelo de despliegue Full Stack híbrido

Si das un paso atrás y superpones las realidades de mercado y negocio sobre el problema, verás cómo algunas organizaciones pueden ver valor en soportar ambos modelos.



Para que este modelo sea viable, debes aplicar restricciones al número de tenants que se les permite operar en un modelo full stack silo. Si la proporción de tenants aislados en silo se vuelve demasiado alta, esto puede socavar toda tu experiencia SaaS.

El modelo de despliegue de Modo Mixto (Mixed Mode)

Aunque es tentador ver los despliegues multi-tenant a través de estos modelos más gruesos, la realidad es que muchos sistemas dependen de un enfoque mucho más fino hacia la multi-tenencia, tomando decisiones de silo y pool a través de toda la superficie de su entorno SaaS.

El modo mixto nos permite mirar las cargas de trabajo dentro de nuestro entorno SaaS y determinar cómo cada uno de los diferentes servicios y recursos debería ser desplegado para cumplir con los requisitos específicos de un caso de uso dado.

Incluso podrían haber casos donde las capas individuales de un servicio podrían tener estrategias de silo/pool separadas. Este es el punto básico que estaba haciendo cuando introduje por primera vez la noción de silo y pool al inicio de este capítulo.

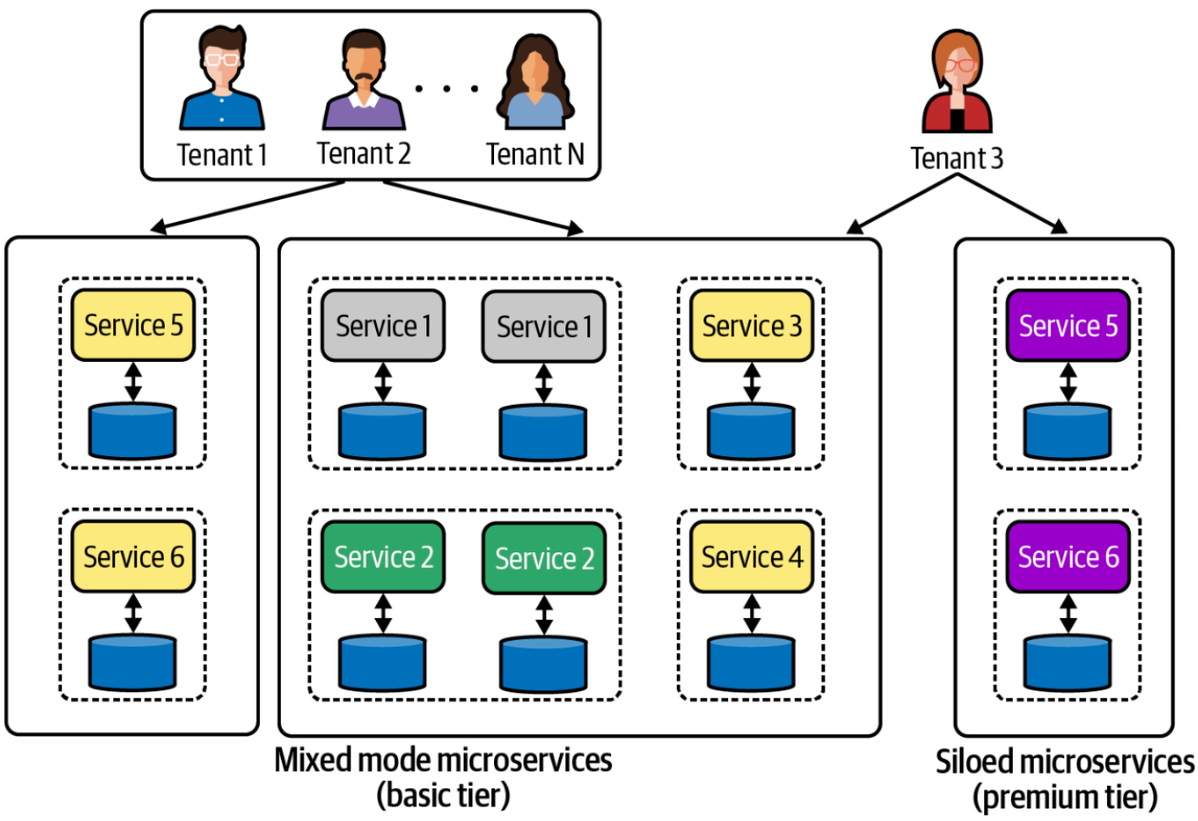


Figure 3-11. A mixed mode deployment model

Abordar tu modelo de despliegue de esta manera más granular proporciona un grado mucho mayor de flexibilidad para ti como arquitecto y para el negocio.

Al soportar modelos silo y pool en todas las capas, tienes la opción de componer la mezcla correcta de experiencias para cumplir con las necesidades de tenant, operacionales y otras que puedan surgir a lo largo de la vida de tu solución.

El modelo de despliegue de Pod (Pod Deployment Model)

También necesito alejarme del enfoque silo/pool y pensar en cómo una aplicación podría necesitar soportar una variación de despliegue que esté más formada por dónde necesita aterrizar, cómo lidia con restricciones ambientales, y cómo podría necesitar transformarse para soportar la escala y alcance de tu negocio SaaS.

> Kubernetes también tiene su propia vista de pods, que es completamente separada de este concepto.

Cuando hablo de pods aquí, estoy hablando de cómo podrías agrupar una colección de tenants en alguna unidad de despliegue. Podría tener alguna motivación técnica, operacional, de cumplimiento, de escala, o de negocio que me empuje hacia un modelo donde pongo tenants en pods individuales, y estos pods se convierten en una unidad de despliegue, gestión y operación para mi negocio SaaS.

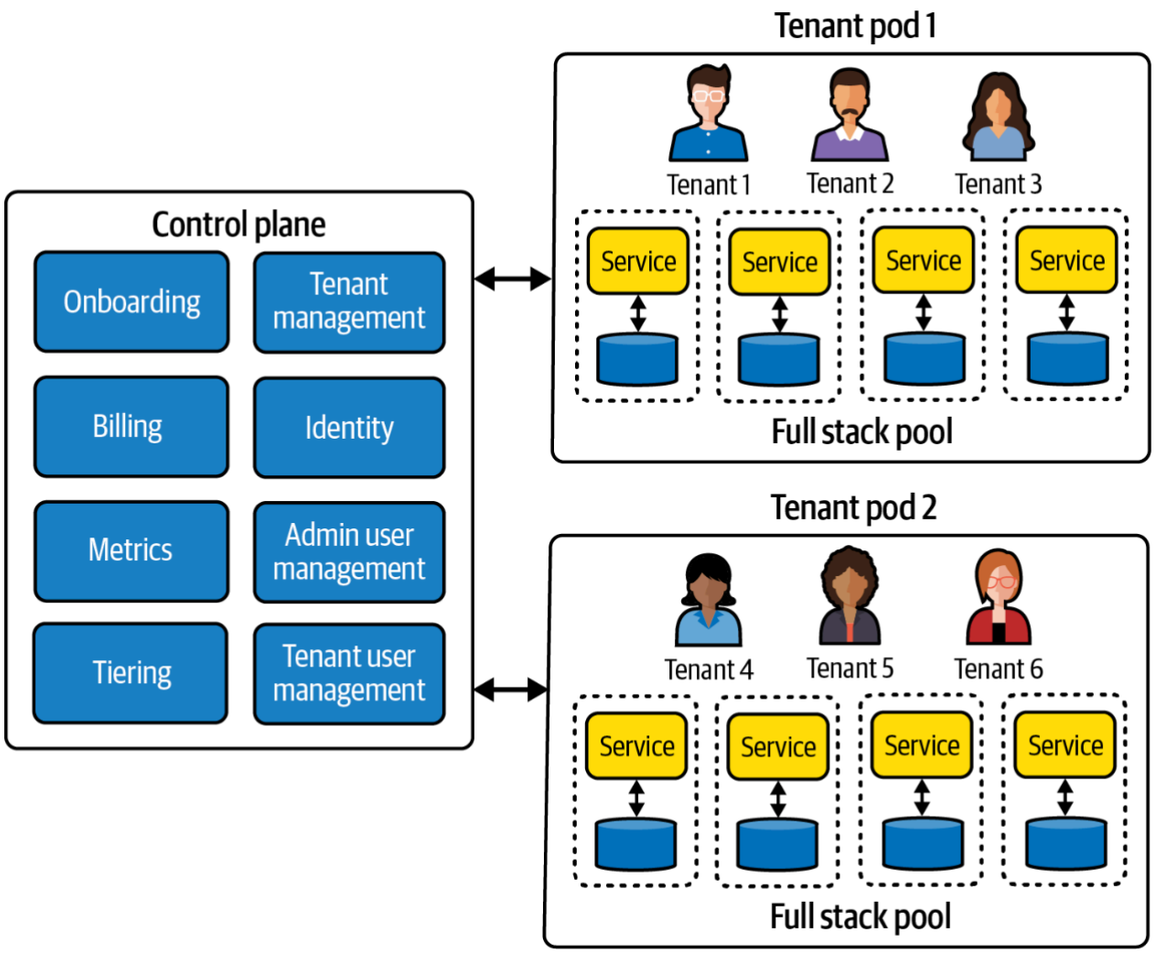


Figure 3-12. A pod deployment model

Estos pods separados traen un grado de complejidad a un entorno SaaS, requiriendo que tu control plane construya los mecanismos para soportar este modelo de distribución.

Cómo los tenants se incorporan, por ejemplo, debe tener en cuenta en qué pod un tenant dado aterrizará. Tu gestión y operaciones también deben ser conscientes de los pods, proporcionando perspectivas de la salud y actividad de cada pod.

Hay varios factores que podrían impulsar la adopción de un modelo de entrega basado en pods. Imagina, por ejemplo, tener un modelo full stack pool ejecutándose en la nube que, en un cierto número de tenants, comienza a exceder los límites de infraestructura de servicios específicos.

Esto también podría ser impulsado por una necesidad de desplegar un producto SaaS en múltiples geografías; requisitos geográficos o consideraciones de rendimiento podrían inclinarte hacia un modelo de despliegue basado en pods, donde diferentes geografías podrían estar ejecutando diferentes pods.

Algunos equipos también pueden usar pods como una estrategia de aislamiento para reducir impactos entre tenants. Esto puede ser motivado por una necesidad de mayores protecciones de condiciones de vecino ruidoso (noisy neighbor), o puede jugar un papel en la historia de seguridad y disponibilidad de un proveedor SaaS.

Adoptar un modelo de pod significa comprometerse a absorber la complejidad adicional y la automatización que te permite soportar y gestionar pods sin tener mecanismos específicos para pods individuales.

Para escalar exitosamente, la configuración y despliegue de estos pods deben estar completamente automatizados a través de tu control plane.

Una dinámica que viene con pods es esta idea de ver la membresía dentro de un pod como algo que puede ser cambiado durante la vida de un tenant. **Sin embargo, esto vendría con un esfuerzo significativo para obtener toda la huella del tenant transferida a otro pod.**