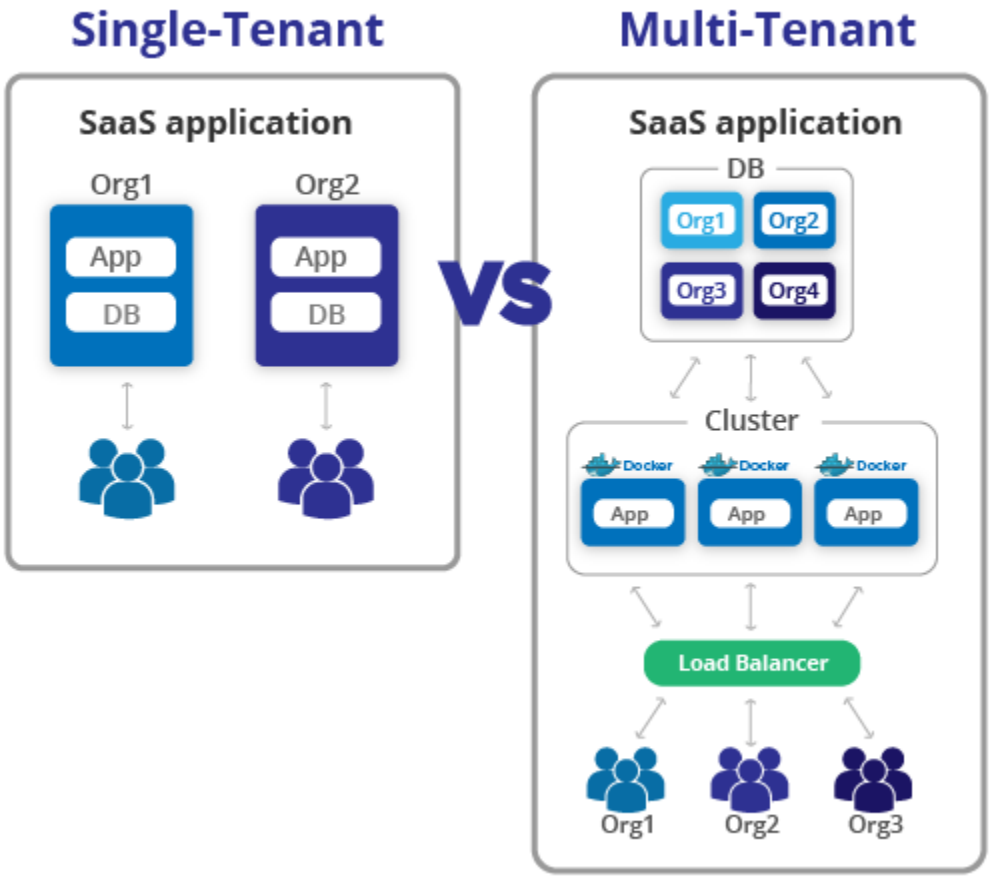
1 SOLUCIÓN SAAS MULTIINQUILINO - CONSIDERACIONES ARQUITECTÓNICAS

1.1 ¿QUÉ ES MULTI-INQUILINOS?

Arquitectura multiinquilino es una arquitectura de software que ejecuta varias instancias individuales del software en un único servidor físico, que sirve a varios inquilinos.

La arquitectura multiinquilino es un ecosistema o modelo, en el que un único entorno puede servir a varios inquilinos utilizando una arquitectura escalable, disponible y resistente. La infraestructura subyacente está completamente compartida, lógicamente aislada y con servicios totalmente centralizados. La arquitectura multiinquilino evoluciona según la organización o el subdominio que se registra en la aplicación SaaS; y es totalmente transparente para el usuario.



1.2 MEJORES PRÁCTICAS GENERALES

- En lugar de crear grandes arquitecturas de aplicaciones monolíticas, a menudo resulta útil crear servicios más pequeños, independientes y de una sola responsabilidad que se puedan unir para

lograr la funcionalidad empresarial general. Estas arquitecturas más pequeñas basadas en microservicios pueden ser más fáciles de administrar y pueden escalar de forma independiente.
- Cree la abstracción en cada capa para que pueda preparar la solución para el futuro, pudiendo cambiar la implementación subyacente sin afectar a las interfaces públicas.
- Defina un proceso de gestión de versiones para habilitar actualizaciones de calidad frecuentes en la solución.
- Mantenga las personalizaciones específicas del inquilino al mínimo e intente crear la mayoría de las características dentro de la propia plataforma.
- Cree una API para la solución o plataforma si necesita integrarse con sistemas de terceros. Utilice roles de IAM, en lugar de utilizar credenciales codificadas de forma rígida dentro de varios componentes de la aplicación.
- Utilice Auto Scaling para escalar y reducir dinámicamente su entorno, según la carga.
- Comparar el rendimiento de la aplicación con las instancias de tamaño correcto y su recuento.
- Valide el modelo operativo.
- Valide el modelo de seguridad.

1.3 BENEFICIOS

- Reducción de los costos de infraestructura de servidores
- Una única fuente de confianza
- Reducciones de costos de desarrollo y tiempo de comercialización

1.3.1 PATRONES DE ARQUITECTURA

1.3.2 Criterios de Decisiones

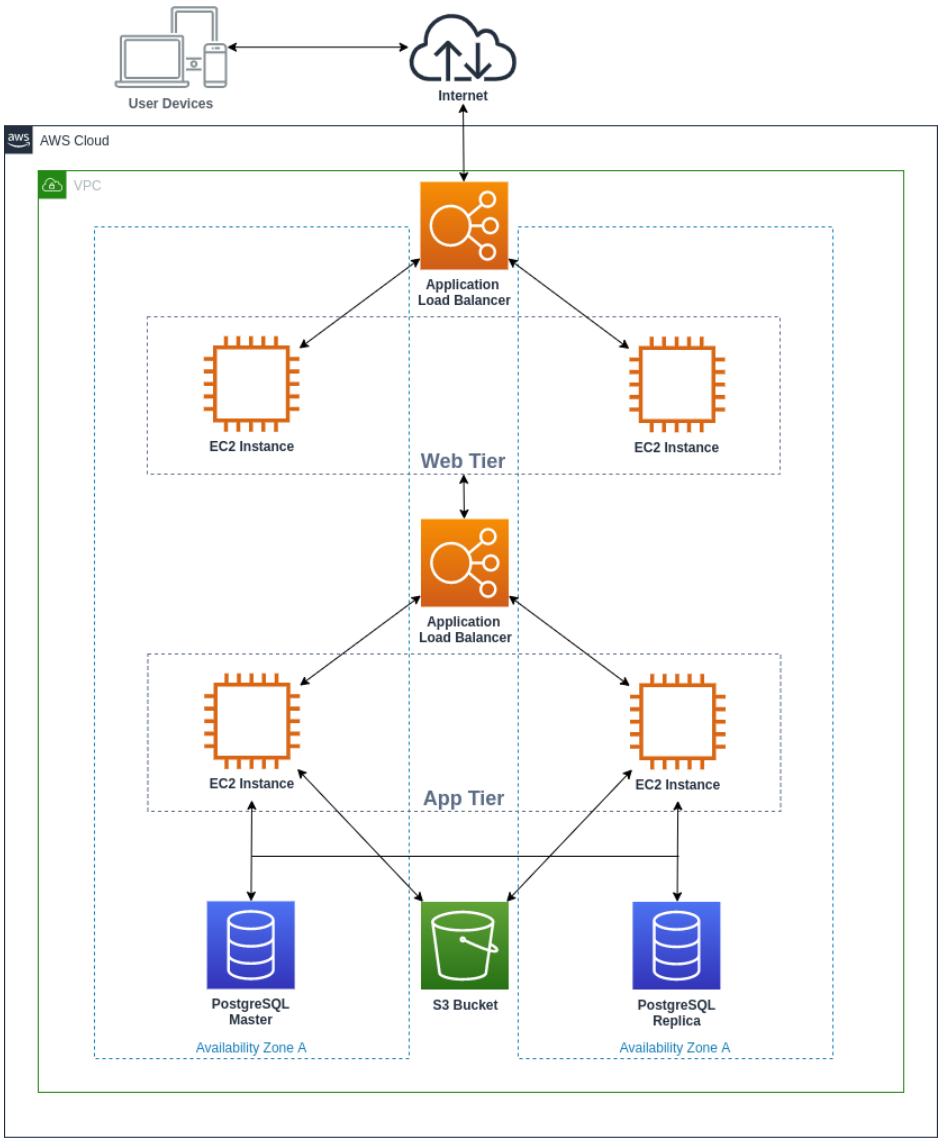
La decisión de elegir un modelo de implementación determinado depende de varios criterios, entre ellos:

- Nivel de segregación entre inquilinos e implementaciones
- Aspectos de escalabilidad de aplicaciones en pilas específicas del inquilino
- Nivel de personalizaciones de aplicaciones específicas del inquilino
- Costo de las operaciones de implementación y los esfuerzos de administración
- Cargo a los inquilinos y aspectos de facturación

1.3.3 Modelos

1.3.3.1 Monolítico SaaS

La arquitectura monolítica no es un enfoque incorrecto, con la excepción de que está desperdiciando recursos masivamente en las diferente capas. Al menos alrededor del 50% y el 70% de su uso de CPU/RAM se desperdicia debido a la naturaleza de la arquitectura monolítica (nube).



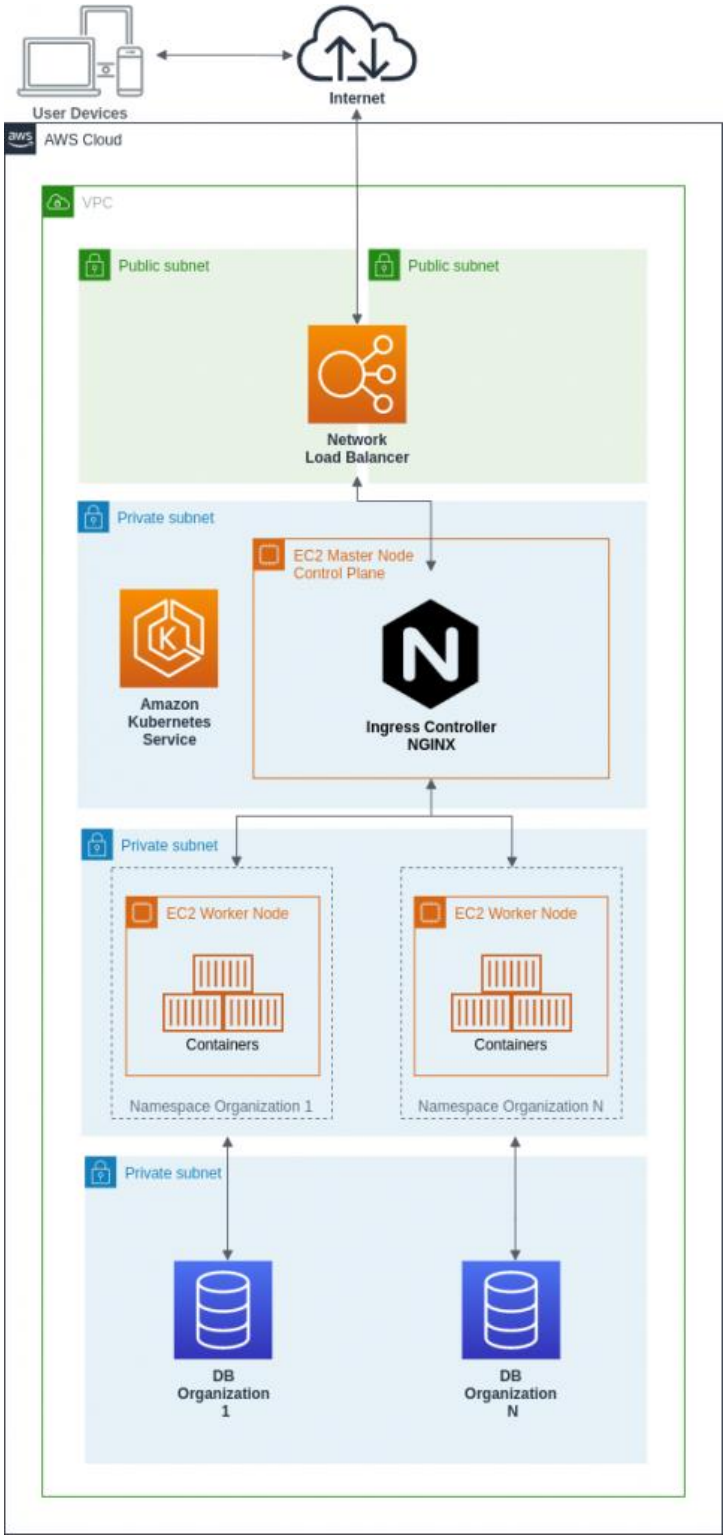
1.3.3.1.1 Pros

- Enfoque fácil de construir
- Configuración mínima
- Base de datos multiinquilino

1.3.3.1.2 Contras

- Desperdicio de recursos
- No muy tolerante a errores por servicio.
- Reimplementación completa de código en el nivel de aplicación.
- Difícil de mantener.
- Tiempo de comercialización lentas
- Restricciones de cumplimiento de HIPAA y PCI

1.3.3.2 SaaS en contenedores



1.3.3.2.1 Pros

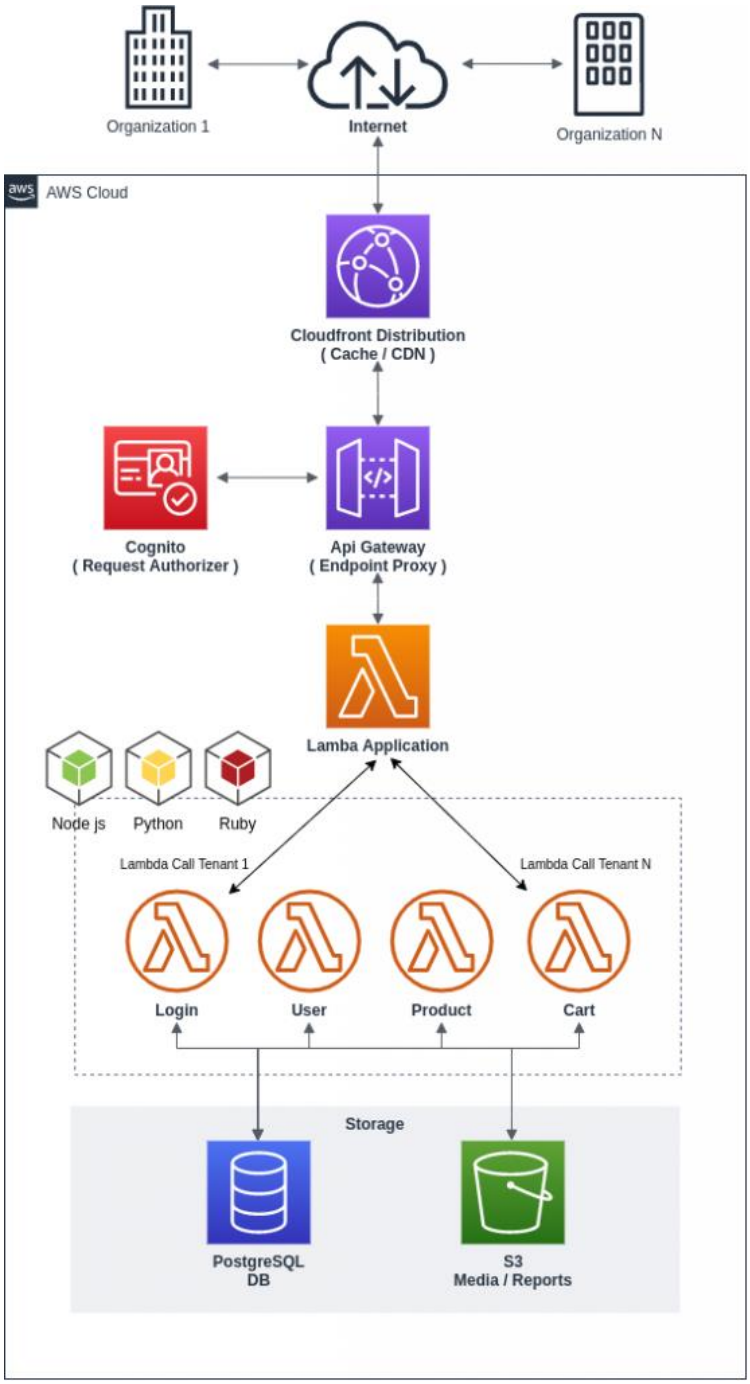
- Excepcional configuración del SaaS personalizada en profundidad.
- Arquitectura acoplada libremente.
- Más fácil de implementar código en producción
- Implementaciones más pequeñas. Mejor agilidad
- Servicio puro y real distribuido.
- Repetibilidad y manejabilidad.
- Mejor utilización de los recursos.

1.3.3.2.2 Contras

- Curva alta de aprendizaje
- Re-arquitectura de la aplicación
- Complejidad para crear la arquitectura de microservicios
- Debe tener cuidado con los servicios propietarios en la nube.

1.3.3.3 SaaS Sin servidor

El sueño de cualquier arquitecto es crear una arquitectura SaaS multiinquilino con un enfoque sin servidor. Ese es un sueño que puede hacerse realidad como arquitecto de DevOps o SaaS, pero agrega especialmente una buena complejidad como compensación. Además, requiere una cantidad razonable de tiempo de colaboración con el equipo de desarrollo, una amplia aplicación de cambios en el código y una mentalidad transformada a una arquitectura y desarrollo sin servidor.



1.3.3.3.1 Pros

- Verdaderamente basado en el consumo por milisegundos. Sin sobre-aprovisionamiento de hardware
- E2E aplicación sin servidor.
- Un microservicio para cada llamada a eventos.
- No hay necesidad de preocuparse por la escalabilidad, el tiempo de inactividad o la disponibilidad. Está integrado.
- Implementación mucho más simplificada.

1.3.3.3.2 Cons

- Curva alta de aprendizaje.
- Añade mucha más compl ejidad a la arquitectura.

1.4 CONSIDERACIONES SOBRE LOS COMPONENTES DE LA ARQUITECTURA

1.4.1 Seguridad y redes

Estrategia para mantener a los inquilinos seguros y aislados unos de otros. Esto puede incluir consideraciones de seguridad como la definición de la segregación en la capa de red/almacenamiento, el cifrado de datos en reposo o en tránsito, la administración segura de claves y certificados e incluso la administración de construcciones de seguridad de nivel de aplicación.

1.4.2 IAM

Se debe establecer una estrategia para autenticar y autorizar a los usuarios a administrar tanto los servicios en la nube como la propia aplicación de SaaS.

1.4.3 Monitoreo, Registro y Rendimiento

El monitoreo debe habilitarse en varias capas, no solo para ayudar a diagnosticar problemas, sino también para habilitar medidas proactivas para evitar problemas en el futuro.

1.4.4 Analíticas

Las soluciones SaaS pueden recopilar una gran cantidad de datos sin ser procesados, incluidos los registros de aplicaciones, los registros de acceso de los usuarios y los datos relacionados con la facturación, que generalmente pueden proporcionar mucha información si se analizan correctamente. Además del análisis orientado a lotes, se pueden realizar análisis en tiempo real para ver qué tipo de acciones están invocando varios inquilinos en la plataforma, o examinar métricas relacionadas con la infraestructura en tiempo real para detectar cualquier comportamiento inesperado y prevenir cualquier problema futuro.

1.4.5 CM & Aprovisionamiento

¡Automatiza, automatiza, automatiza! IAC, automatización de flujos de CI/CD, uso de herramientas de automatización.

1.4.6 Tagging

Se debe establecer una estrategia de etiquetado para ayudar a administrar instancias, imágenes y otros recursos, asignar sus propios metadatos a cada recurso en forma de etiquetas.

1.4.7 Metering & Charging

Establecer la segregación de costos entre los inquilinos en función de su uso. Desde la perspectiva de los recursos de la plataforma, el etiquetado puede ser un gran recurso para ayudar a separar el uso a nivel macro. Sin embargo, para la mayoría de las soluciones SaaS, se necesitan mayores controles para la supervisión del uso, por lo que se recomienda crear un módulo de facturación personalizado propio según sea necesario.

1.4.8 Código

- SaaS basado en URL por cada inquilino.

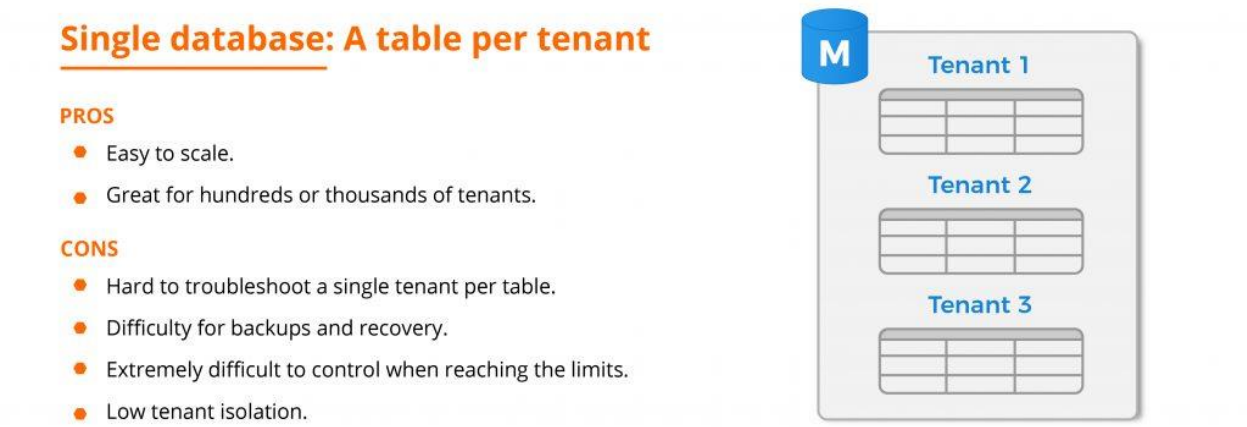
- Administración de certificados SSL
- Sistema de caching
- Configuración segura del servidor web
- CDN Front
- Diseño de microservicios

1.4.9 Data

1.4.9.1 Una Sola Base de Datos

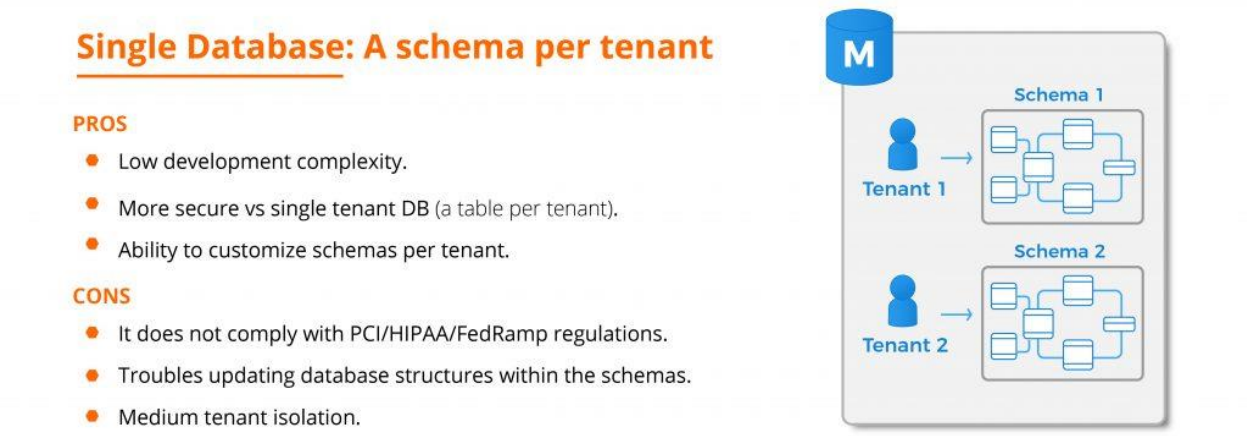
1.4.9.1.1 Una tabla por inquilino

Modelo agrupado. Consiste en aprovechar una tabla por cada organización dentro de un esquema de base de datos.



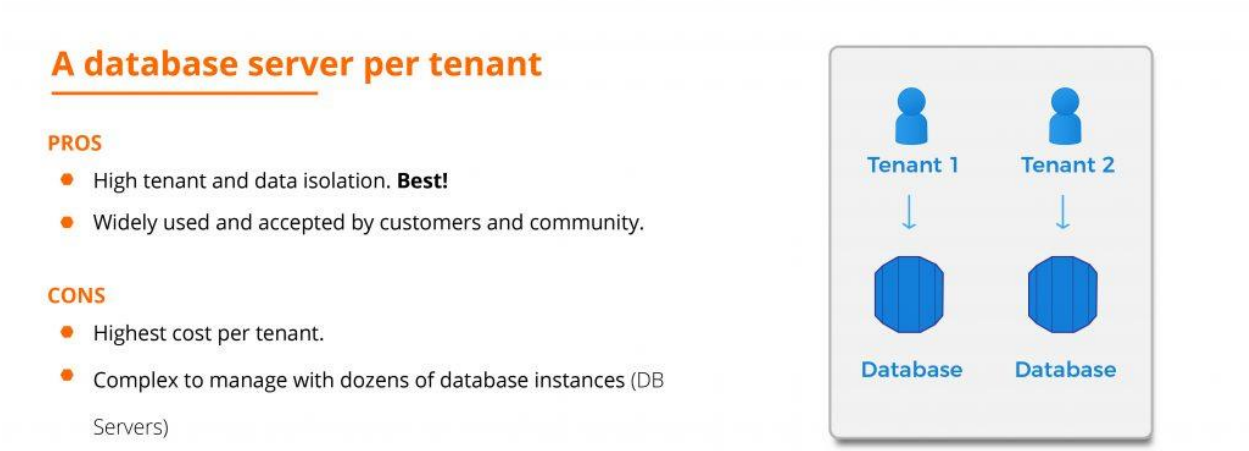
1.4.9.1.2 Un Esquema por Inquilino

Modelo de puente. Este enfoque de base de datos multiinquilino sigue siendo muy rentable y más seguro que la tenencia puro (modelo agrupado de bases de datos), ya que hay una sola base de datos, con la excepción del aislamiento del esquema de base de datos por inquilino.



1.4.9.1.3 Un Servidor de Base de Datos por Inquilino

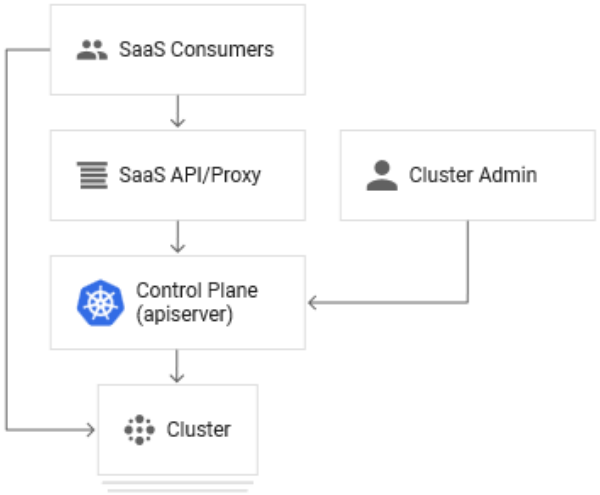
Modelo silo. Esta técnica es significativamente más costosa que el resto de arquitecturas de bases de datos multiinquilino, pero cumple con las regulaciones de seguridad; lo mejor para el rendimiento, la escalabilidad y el aislamiento de datos.



1.5 Consideraciones sobre la Orquestación de Contenedores con Kubernetes

Un clúster multiinquilino es compartido por varios usuarios o cargas de trabajo que se conocen como "inquilinos". Los operadores de clústeres multiinquilino deben aislar a los inquilinos entre sí para minimizar el daño que un inquilino malintencionado puede hacer al clúster y a otros inquilinos. Además, los recursos del clúster deben asignarse de forma justa entre los inquilinos.

Un clúster global de Kubernetes SaaS Multi-tenant podría parecer:



Los inquilinos del clúster de un proveedor SaaS son las instancias por cliente de la aplicación y el plano de control del SaaS. Para aprovechar las directivas de ámbito de namespaces, las instancias de

aplicación deben organizarse en sus propios namespaces, al igual que los componentes del plano de control de SaaS. Los usuarios finales no pueden interactuar directamente con el plano de control de Kubernetes, utilizan la interfaz de SaaS en su lugar, que a su vez interactúa con el plano de control de Kubernetes.

1.5.1 Multi-inquilino clúster

1.5.1.1 Consideraciones

1.5.1.1.1 Configuración organizativa

- Defina la jerarquía de recursos.
- Cree carpetas basadas en su jerarquía organizativa y sus necesidades ambientales.
- Cree proyectos de host y servicio para sus clústeres e inquilinos.

1.5.1.1.2 IAM

- Identifique y cree un conjunto de grupos para las organizaciones.
- Asigne usuarios y políticas de IAM a los grupos.
- Refinar el acceso de inquilino con roles de ámbito de namespaces y enlaces de roles.
- Conceda acceso de administrador de inquilinos para administrar los usuarios de inquilinos.

1.5.1.1.3 Redes

- Cree redes de VPC compartida por entorno para las redes de inquilinos y clústeres.

1.5.1.1.4 Alta disponibilidad y fiabilidad

- Cree un proyecto de administración de clúster por clúster para reducir los impactos adversos en los clústeres.
- Cree el clúster como un clúster privado.
- Asegúrese de que el plano de control del clúster sea regional.
- Intervalo de nodos para el clúster en al menos tres zonas.
- Habilite el escalado automático de clústeres y el escalado automático de Pod.
- Especifique las ventanas de mantenimiento que se producirán durante las horas de menor actividad.
- Cree un equilibrador de carga HTTP(s) para permitir una entrada única por clúster multiinquilino.

1.5.1.1.5 Seguridad

- Cree namespaces para proporcionar aislamiento entre los inquilinos que están en el mismo clúster.
- Cree directivas de red para restringir la comunicación entre pods.
- Mitigue las amenazas ejecutando cargas de trabajo en un entorno limitado.
- Cree directivas de seguridad de Pod para restringir el funcionamiento de los pods en el clúster.
- Habilite Workload Identity para administrar las cuentas de servicio y el acceso a Kubernetes.
- Habilite las redes autorizadas para restringir el acceso al plano de control.

1.5.1.1.6 Logging y Monitoreo

- Aplique cuotas de recursos para cada namespace.
- Realice un seguimiento de las métricas de uso con la medición de uso.

- Configure el registro específico del inquilino con la Kubernetes Engine Monitoring.
- Configure monitoreo específico del inquilino.

1.5.1.2 Ventajas

Las ventajas mas destancates del uso de un Kubernetes clúster para in Muli-inquilino SaaS:

- Reducción de la sobrecarga de administración
- Reducción de la fragmentación de recursos
- No es necesario esperar por la creación del clúster para los nuevos inquilinos