

Contents

[Documentação do Banco de Dados do Azure para PostgreSQL](#)

[Visão geral](#)

[O que é o Banco de Dados do Azure para PostgreSQL?](#)

[Qual opção de implantação do PostgreSQL deve ser escolhida?](#)

[Atualizações de serviço](#)

[Expiração do certificado raiz SSL](#)

[Documentação compartilhada](#)

[Implantação de banco de dados](#)

[Ações do GitHub](#)

[Migração](#)

[Migrar dados com pg_dump e pg_restore](#)

[Migrar dados com pg_dump e psql](#)

[Migração com tempo de inatividade mínimo usando o DMS](#)

[Migrar do Oracle](#)

[Política de controle de versão](#)

[Atualizações de versão principal](#)

[Problemas e limitações conhecidos](#)

[Servidor único](#)

[Documentação do servidor único](#)

[Visão geral](#)

[Inícios rápidos](#)

[Criar um servidor](#)

[Portal do Azure](#)

[CLI do Azure](#)

[Comando up da CLI do Azure](#)

[Azure PowerShell](#)

[Modelo de ARM](#)

[Conectar e consultar](#)

[Python](#)

[Node.js](#)

[Java](#)

[Java com Spring Data JDBC](#)

[Java com Spring Data JPA](#)

[Java com Spring Data R2DBC](#)

[Ruby](#)

[PHP](#)

[.NET](#)

[Go](#)

[Rust](#)

[Tutoriais](#)

[Criar um banco de dados](#)

[Portal do Azure](#)

[CLI do Azure](#)

[Azure PowerShell](#)

[Compilar e implantar o aplicativo Web Python \(Django\)](#)

[Monitorar e ajustar](#)

[Exemplos](#)

[CLI do Azure](#)

[Itens internos do Azure Policy](#)

[Conceitos](#)

[Servidores](#)

[Versões com suporte](#)

[Extensões do PostgreSQL](#)

[Limites](#)

[Entender os tipos de preço](#)

[Tipos de preço](#)

[Pagar antecipadamente por capacidade reservada](#)

[Segurança](#)

[Visão geral da segurança](#)

[Configurar SSL](#)

[Arquitetura de conectividade](#)

Proteção contra ameaças com o Azure Defender

Autenticação do Azure AD

Criptografia de dados

Criptografia dupla de infraestrutura

Controles de segurança do Azure Policy

Linha de base de segurança

Rede

Regras de firewall

Rede virtual

Link privado

Manutenção planejada

Continuidade de negócios

Introdução à continuidade dos negócios

Alta disponibilidade

Backup e restauração

Monitorar e ajustar

Monitorar e ajustar

Logs

Logs de auditoria

Repositório de consultas

Repositório de consultas

Cenários de uso do repositório de consultas

Melhores práticas para o repositório de consultas

Análise de desempenho de consultas

Recomendações do desempenho

Recomendações do Assistente do Azure

Desenvolvimento de aplicativos

Práticas recomendadas

Bibliotecas de conexão

Resiliência de conexão

Replicação

Réplicas de leitura

Decodificação lógica

Guias de instruções

Gerenciar e escalar

Portal do Azure

CLI do Azure

Reiniciar servidor

Portal do Azure

CLI do Azure

Azure PowerShell

Regras de firewall

Portal do Azure

CLI do Azure

Guia Conectar e consultar

Restaurar servidor

Portal do Azure

CLI do Azure

Azure PowerShell

Restaurar um servidor descartado

Habilitar autenticação

Criar usuários

Configurar a integração do Azure AD

Conectar-se com Identidade Gerenciada

Otimizar

Inserções em massa

Vácuo automático

Coleta das estatísticas de consulta

Estratégia da tabela de notificação do sistema

Configurar TLS

Cadeias de conexão

Configurar parâmetros do servidor

Portal do Azure

CLI do Azure

Azure PowerShell

Armazenamento de crescimento automático

Portal do Azure

CLI do Azure

Azure PowerShell

Logs de acesso

Portal do Azure

CLI do Azure

Monitoramento

Criar alertas nas métricas

Solucionar problemas

Solucionar erros de conexão

Solucionar problemas de criptografia de dados

Rede virtual

Portal do Azure

CLI do Azure

Link privado

Portal do Azure

CLI do Azure

Criptografia de dados

Portal do Azure

CLI do Azure

Validação de criptografia de dados

Criptografia dupla de infraestrutura

Configurar a criptografia dupla

Negar acesso à rede pública

Portal do Azure

Replicação

Portal do Azure

CLI do Azure, API REST

Azure PowerShell

Mover entre regiões

- Portal do Azure
- Servidor flexível (versão prévia)
 - Documentação do servidor flexível
 - Visão geral
 - Notas de versão
 - Inícios rápidos
 - Criar banco de dados e servidor
 - Portal do Azure
 - CLI do Azure
 - Modelo de ARM
 - Conectar e consultar
 - CLI do Azure
 - Python
 - Java
 - .NET
- Tutoriais
 - Criar usando o AKS
 - Implantar aplicativo Python no Kubernetes
 - Criar usando os Serviços de Aplicativos
 - Criar um aplicativo Web em uma rede virtual
 - Implantar um aplicativo Django no Serviço de Aplicativo
- Conceitos
 - Servidores
 - Versões com suporte
 - Computação e armazenamento
 - Rede
 - Regras de firewall
 - Segurança
 - Limites
 - Extensões
 - Manutenção agendada
 - Pooling de conexões (PgBouncer)

[Parâmetros do Servidor](#)

[Continuidade de negócios](#)

[Visão geral da continuidade dos negócios](#)

[Backup e restauração](#)

[Alta disponibilidade](#)

[Monitorar e ajustar](#)

[Visão geral de monitoramento e ajuste](#)

[Logs](#)

[Logs de auditoria](#)

[Desempenho inteligente](#)

[Visão geral do Repositório de Consultas](#)

[Cenários de uso do Repositório de Consultas](#)

[Práticas recomendadas para o Repositório de Consultas](#)

[Replicação](#)

[Guias de instruções](#)

[Gerenciar um servidor](#)

[Portal do Azure](#)

[CLI do Azure](#)

[Guia Conectar e consultar](#)

[Manutenção agendada](#)

[Portal do Azure](#)

[Rede](#)

[Acesso privado \(Integração VNet\)](#)

[Portal do Azure](#)

[CLI do Azure](#)

[Acesso público \(endereços IP permitidos\)](#)

[Portal do Azure](#)

[CLI do Azure](#)

[TLS/SSL/SCRAM](#)

[Conectar-se usando TLS/SSL](#)

[Conectar-se usando SCRAM](#)

[Configurar parâmetros do servidor](#)

- [Portal do Azure](#)
- [CLI do Azure](#)
- [Dimensionar um servidor](#)
 - [Portal do Azure](#)
 - [Reiniciar um servidor](#)
 - [Portal do Azure](#)
 - [CLI do Azure](#)
 - [Restaurar um servidor](#)
 - [Portal do Azure](#)
 - [CLI do Azure](#)
 - [Parar/iniciar um servidor](#)
 - [Portal do Azure](#)
 - [CLI do Azure](#)
 - [Gerenciar alta disponibilidade](#)
 - [Portal do Azure](#)
 - [CLI do Azure](#)
 - [Monitoramento](#)
 - [Criar alertas em métricas usando o portal](#)
 - [Configurar e acessar logs](#)
- [Hiperescala \(Citus\)](#)
 - [Documentação da Hiperescala \(Citus\)](#)
 - [Visão geral](#)
 - [O que é a Hiperescala \(Citus\)?](#)
 - [Recursos na visualização](#)
 - [Início Rápido](#)
 - [Criar grupo de servidores](#)
 - [Camada básica \(versão prévia\)](#)
 - [Camada padrão](#)
 - [Tutoriais](#)
 - [Criar um grupo de servidor](#)
 - [Modelar e carregar dados](#)
 - [Dados de fragmentos em nós de trabalho](#)

Criar um banco de dados multilocatário

Criar um painel em tempo real

Conceitos

Camada básica

Dados distribuídos

Nós e tabelas

Determinar o tipo de aplicativo

Escolher uma coluna de distribuição

Colocação de tabela

Segurança

Regras de firewall

Configurar SSL

Linha de base de segurança

Manutenção agendada

Continuidade de negócios

Backup e restauração

Alta disponibilidade

Replicação

Pool de conexões

Monitorar e ajustar

Monitorar e ajustar

Logs de auditoria

Armazenamento em coluna

Opções de configuração

Limites e limitações

Pagar antecipadamente por capacidade reservada

Versões do PostgreSQL

Extensões do PostgreSQL

Guias de instruções

Tamanho do grupo de servidores

Escolher tamanho inicial

Escalar grupo de servidores

- [Reequilibrar fragmentos](#)
- [Regras de firewall](#)
- [Portal do Azure](#)
- [Criar usuários](#)
- [Dados distribuídos](#)
 - [Determinar o tamanho da tabela](#)
 - [Distribuir e modificar tabelas](#)
- [Alta disponibilidade](#)
- [Replicação](#)
 - [Portal do Azure](#)
- [Manutenção agendada](#)
 - [Portal do Azure](#)
- [Reiniciar grupo de servidores](#)
- [Monitoramento](#)
 - [Criar alertas nas métricas](#)
 - [Acessar logs do banco de dados](#)
- [Restaurar grupo de servidores](#)
 - [Portal do Azure](#)
- [Atualizações de versão principal](#)
- [Solucionar problemas](#)
 - [Solucionar erros de conexão](#)
 - [Solucionar problemas de acesso somente leitura](#)
 - [Consultas de diagnóstico úteis](#)
- [Referência de API](#)
 - [Funções da API SQL](#)
 - [Parâmetros do Servidor](#)
 - [Tabelas do sistema](#)
- [Referência](#)
 - [CLI do Azure](#)
 - [API REST](#)
 - [Modelo do Resource Manager](#)
 - [Itens internos do Azure Policy](#)

Recursos

[Desenvolva suas habilidades com o Microsoft Learn](#)

[Modelos de implantação](#)

[Roteiro do Azure](#)

[Preços](#)

[Página de perguntas de P e R da Microsoft](#)

[Stack Overflow](#)

[vídeos](#)

[Vídeos do produto](#)

[Como fazer uma série de vídeos](#)

[Saiba mais sobre os benefícios e recursos do produto](#)

[Migrar seu aplicativo PostgreSQL para o Azure](#)

[Compilar aplicativos inteligentes com Serviços Cognitivos](#)

[Conectar contêineres usando OSBA](#)

[Conectar dados do aplicativo ao Power BI](#)

[Fórum de comentários](#)

[Disponibilidade de região](#)

[Opções de suporte](#)

[Relatos de clientes](#)

[Parceiros](#)

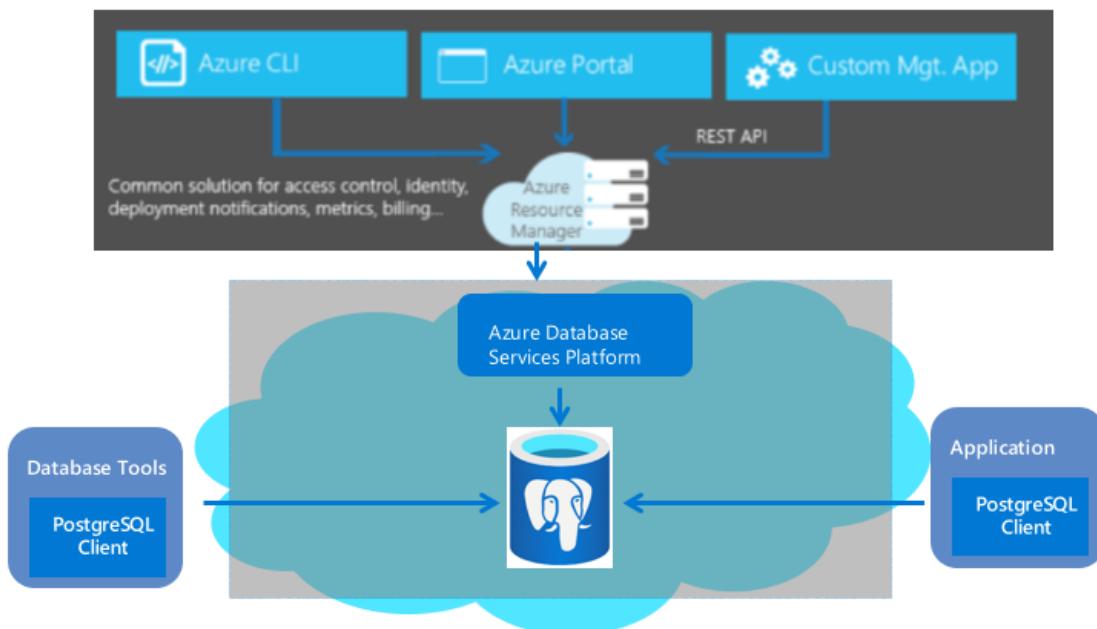
[Guia de migração de banco de dados](#)

O que é o Banco de Dados do Azure para PostgreSQL?

21/05/2021 • 4 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço de banco de dados relacional no Microsoft Cloud com base no mecanismo de banco de dados [PostgreSQL Community Edition](#) (disponível sob a licença GPLv2). O Banco de Dados do Azure para PostgreSQL fornece:

- Alta disponibilidade interna.
- Proteção de dados usando backups automáticos e restauração pontual por até 35 dias.
- Manutenção automatizada para hardware, sistema operacional e mecanismo de banco de dados subjacentes para manter o serviço seguro e atualizado.
- Desempenho previsível, com preços pré-pagos inclusivos.
- Dimensionamento elástico em segundos.
- Segurança de nível corporativo e conformidade líder do setor para proteger dados confidenciais em repouso e em movimento.
- Monitoramento e automação para simplificar o gerenciamento e o monitoramento para implantações em larga escala.
- Experiência de suporte líder do setor.



Esses recursos não precisam de quase nenhuma administração e todos são fornecidos sem nenhum custo adicional. Eles permitem que você se concentre no método RAD e em acelerar seu tempo de colocação no mercado, em vez de alocar tempo e recursos preciosos ao gerenciamento de máquinas virtuais e de infraestrutura. Além disso, você pode continuar desenvolvendo seu aplicativo com a plataforma e as ferramentas de software livre de sua escolha e pode fornecê-lo com a velocidade e a eficiência que sua empresa exige, tudo isso sem precisar aprender novas habilidades.

Modelos de implantação

O Banco de Dados do Azure para PostgreSQL desenvolvido com o PostgreSQL community edition está disponível em três modos de implantação:

- Servidor único
- Servidor Flexível (versão prévia)
- Hiperescala (Citus)

Banco de Dados do Azure para PostgreSQL – Servidor único

O Banco de Dados do Azure para PostgreSQL Single Server é um serviço de banco de dados totalmente gerenciado com requisitos mínimos para personalizações do banco de dados. A plataforma de servidor único é projetada para administrar a maioria das funções de gerenciamento de banco de dados, como aplicação de patches, backups, alta disponibilidade, controle e segurança com mínima configuração do usuário. A arquitetura é otimizada para alta disponibilidade interna com disponibilidade de 99,99% em uma zona de disponibilidade única. Ela dá suporte à versão da comunidade do PostgreSQL 9.5, 9.6, 10 e 11. O serviço está em disponibilidade geral em diversas [regiões do Azure](#).

A opção de implantação Servidor Único oferece três tipos de preço: Básico, Uso Geral e Otimizado para Memória. Cada tipo oferece recursos diferentes para dar suporte a suas cargas de trabalho do banco de dados. Você pode criar seu primeiro aplicativo em um banco de dados pequeno por alguns dólares por mês e então ajustar a escala para atender às necessidades da sua solução. A escalabilidade dinâmica permite que o banco de dados responda de forma transparente a mudanças rápidas nos requisitos de recursos. Você paga apenas pelos recursos de que precisa, e somente quando precisa deles. Veja [Tipos de preço](#) para obter detalhes.

Servidores únicos são mais adequados para aplicativos nativos na nuvem projetados para lidar com aplicação de patch automatizada sem a necessidade de controle granular sobre o agendamento de aplicação de patch e definições de configuração de PostgreSQL personalizadas.

Para obter uma visão geral detalhada do modo de implantação de servidor único, confira [visão geral de servidor único](#).

Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão prévia)

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL é um serviço de banco de dados totalmente gerenciado projetado para fornecer controle e flexibilidade mais granulares nas funções de gerenciamento de banco de dados e definições de configuração. Em geral, o serviço fornece mais flexibilidade e personalizações conforme os requisitos do usuário. A arquitetura de servidor flexível permite que os usuários optem por alta disponibilidade dentro de uma zona de disponibilidade única e entre várias zonas de disponibilidade. O Servidor Flexível fornece controles de otimização de custo melhores com a capacidade de parar/iniciar o servidor e a camada de computação expansível, ideal para cargas de trabalho que não precisam de capacidade de computação completa continuamente. O serviço atualmente dá suporte à versão da comunidade do PostgreSQL 11 e 12 com planos para adicionar versões mais recentes em breve. O serviço está em versão prévia pública, disponível em diversas regiões do Azure.

Os servidores flexíveis são mais adequados para

- Desenvolvimentos de aplicativos que exigem controle e personalizações melhores.
- Controles de otimização de custos com capacidade de parar/iniciar o servidor.
- Alta disponibilidade com redundância de zona
- Janelas de manutenção gerenciadas

Para obter uma visão geral detalhada do modo de implantação de servidor flexível, confira [visão geral do servidor flexível](#).

Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

A opção Hiperescala (Citus) escala horizontalmente as consultas em vários computadores usando a fragmentação. Seu mecanismo de consulta faz a correspondência entre consultas SQL recebidas nesses servidores para obter respostas mais rápidas em grandes conjuntos de dados. Ele atende aplicativos que exigem maior escala e desempenho, geralmente cargas de trabalho que estão se aproximando, ou já excederam, 100 GB

de dados.

A opção de implantação Hiperescala (Citus) oferece:

- Dimensionamento horizontal entre vários computadores usando a fragmentação
- Consultar a paralelização nesses servidores para obter respostas mais rápidas em grandes conjuntos de dados
- Um excelente suporte para aplicativos multilocatário, análise operacional em tempo real e cargas de trabalho transacionais de alta taxa de transferência

Os aplicativos criados para o PostgreSQL podem executar consultas distribuídas em Hiperescala (Citus) com [bibliotecas de conexão](#) padrão e alterações mínimas.

Próximas etapas

Saiba mais sobre os três modos de implantação para o Banco de Dados do Azure para PostgreSQL e escolha as opções corretas conforme suas necessidades.

- [Servidor único](#)
- [Servidor Flexível](#)
- Hiperescala (Citus)

Escolher a opção certa do servidor PostgreSQL no Azure

21/05/2021 • 7 minutes to read

Com o Azure, as suas cargas de trabalho do Servidor PostgreSQL podem ser executadas como uma PaaS (plataforma como serviço) hospedada ou em uma IaaS (infraestrutura como serviço) de máquina virtual hospedada. A PaaS tem várias opções de implantação, cada uma com várias camadas de serviço. Ao escolher entre IaaS e PaaS você precisa decidir se deseja gerenciar o banco de dados, aplicar patches e fazer backups ou se deseja delegar essas operações ao Azure.

Ao tomar essa decisão, considere as três opções a seguir na PaaS ou, como alternativa, em execução nas VMs do Azure (IaaS)

- [Servidor Único do banco de dados do Azure para PostgreSQL](#)
- [Servidor Flexível do banco de dados do Azure para PostgreSQL](#)
- [Hiperescala \(Citus\) do banco de dados do Azure para PostgreSQL](#)

A opção **PostgreSQL em VMs do Azure** se enquadra na categoria do setor de IaaS. Com esse serviço, você pode executar o Servidor PostgreSQL dentro de uma máquina virtual totalmente gerenciada na plataforma de nuvem do Azure. Todas as versões e edições recentes do PostgreSQL podem ser instaladas em uma máquina virtual IaaS. A diferença mais significativa do Banco de Dados do Azure para PostgreSQL é que o PostgreSQL em VMs do Azure oferece controle sobre o mecanismo de banco de dados. No entanto, esse controle é acompanhado da responsabilidade de gerenciar as VMs e muitas tarefas de DBA (administração de banco de dados). Essas tarefas incluem manutenção e aplicação de patches de servidores de banco de dados, recuperação de banco de dados e design de alta disponibilidade.

As diferenças principais entre essas opções estão listadas na tabela a seguir:

ATTRIBUTO	POSTGRES EM VMs DO AZURE	POSTGRESQL COMO PAAS
SLA de Disponibilidade	– 99,99% com conjuntos de disponibilidade – 99,95% com VMs únicas	– Servidor Único – 99,99% – Servidor Flexível – Indisponível durante a Versão Prévia – Hiperescala (Citus) – 99,95% (quando a alta disponibilidade está habilitada)
Aplicação de patch no sistema operacional e no PostgreSQL	– Gerenciado pelo cliente	– Servidor Único – Automático – Servidor Flexível – Automático com janela gerenciada opcional do cliente – Hiperescala (Citus) – Automático
Alta disponibilidade	– Os clientes arquitetam, implementam, testam e mantêm a alta disponibilidade. As funcionalidades podem incluir clustering, replicação etc.	– Servidor Único: interno – Servidor Flexível: interno – Hiperescala (Citus): criado com o modo em espera
Redundância de Zona	– As VMs do Azure podem ser configuradas para serem executadas em diferentes zonas de disponibilidade. Para uma solução local, os clientes precisam criar, gerenciar e manter o próprio data center secundário.	– Servidor Único: Não – Servidor Flexível: Sim – Hiperescala (Citus): Não

ATRIBUTO	POSTGRES EM VMS DO AZURE	POSTGRESQL COMO PAAS
Cenário Híbrido	– Gerenciado pelo cliente	– Servidor Único: réplica de leitura – Servidor Flexível: indisponível durante a Versão Prévia – Hiperescala (Citus): Não
Backup e Restauração	– Gerenciado pelo Cliente	– Servidor Único: interno com a configuração do usuário para área geográfica e local – Servidor Flexível: interno com a configuração do usuário no armazenamento com redundância de zona – Hiperescala (Citus): interno
Monitoramento de Operações de Banco de Dados	– Gerenciado pelo Cliente	– Servidor Único, Servidor Flexível e Hiperescala (Citus): todos oferecem aos clientes a capacidade de definir alertas na operação de banco de dados e agir quando os limites estiverem sendo alcançados.
Proteção Avançada contra Ameaças	– Os clientes precisam criar essa proteção para eles mesmos.	– Servidor Único: Sim – Servidor Flexível: indisponível durante a Versão Prévia – Hiperescala (Citus): Não
Recuperação de Desastre	– Gerenciado pelo Cliente	– Servidor Único: backup com redundância geográfica e réplica com leitura geográfica – Servidor Flexível: indisponível durante a Versão Prévia – Hiperescala (Citus): Não
Desempenho Inteligente	– Gerenciado pelo Cliente	– Servidor Único: Sim – Servidor Flexível: indisponível durante a Versão Prévia – Hiperescala (Citus): Não

TCO (custo total de propriedade)

Frequentemente, o TCO é principal consideração que determina a melhor solução para hospedar os seus bancos de dados. Isso é verdadeiro tanto se você for uma startup com pouco dinheiro como uma equipe em uma empresa estabelecida que opere sob restrições de orçamento rígidas. Esta seção descreve as noções básicas de cobrança e licenciamento no Azure, pois elas se aplicam ao Banco de Dados do Azure para PostgreSQL e ao PostgreSQL em VMs do Azure.

Cobrança

O Banco de Dados do Azure para PostgreSQL está disponível atualmente como um serviço em vários níveis com preços diferentes para recursos. Todos os recursos são cobrados por hora a uma taxa fixa. Para obter as informações mais recentes sobre as camadas de serviço, tamanhos de computação e valores de armazenamento com suporte no momento, confira a [página de preços](#). Você pode ajustar dinamicamente as camadas de serviço e os tamanhos de computação para corresponder às diversas necessidades de taxa de transferência do seu aplicativo. Você será cobrado pelo tráfego de Internet de saída em [taxas de transferência de dados regulares](#).

Com o Banco de Dados do Azure para PostgreSQL, a Microsoft configura automaticamente, aplica patches e atualiza o software de banco de dados. Essas ações automatizadas reduzem os custos de administração. Além disso, o Banco de Dados do Azure para PostgreSQL tem funcionalidades de [link de backup automatizado](#). Essas funcionalidades ajudam você a obter economia significativa, principalmente quando você tem um grande número de bancos de dados. Por outro lado, com o PostgreSQL em VMs do Azure, você pode escolher e executar qualquer versão do PostgreSQL. No entanto, você precisa pagar pela VM provisionada, pelo custo de armazenamento associado aos dados, backup, monitoramento de dados e armazenamento de logs, bem como os custos do tipo de licença do PostgreSQL específico usado (se houver).

O Banco de Dados do Azure para PostgreSQL fornece alta disponibilidade interna para qualquer tipo de interrupção no nível de nó e, ao mesmo tempo, mantém a garantia de 99,99% de SLA para o serviço. No entanto, para alta disponibilidade do banco de dados nas VMs, use as opções de alta disponibilidade como [Replicação de Streaming](#) que estão disponíveis em um banco de dados PostgreSQL. Usar uma opção de alta disponibilidade com suporte não fornece um SLA adicional. Mas permite que você obtenha mais de 99,99% de disponibilidade de banco de dados com custo adicional e sobrecarga administrativa.

Para obter mais informações sobre preços, confira os seguintes artigos:

- [Preço do Banco de Dados do Azure para PostgreSQL](#)
- [Preço da máquina virtual](#)
- [Calculadora de preços do Azure](#)

Administração

Para muitas empresas, a redução da complexidade de administração é tão importante na decisão de fazer a transição para um serviço de nuvem quanto o custo.

Com a IaaS, a Microsoft:

- Administra a infraestrutura subjacente.
- Fornece aplicação de patch automatizada para o hardware e o sistema operacional subjacentes

Com a PaaS, a Microsoft:

- Administra a infraestrutura subjacente.
- Fornece a aplicação automatizada de patch do hardware, do sistema operacional e do mecanismo de banco de dados subjacentes.
- Gerencia a alta disponibilidade do banco de dados.
- Executa backups automaticamente e replica todos os dados para fornecer recuperação de desastre.
- Criptografa os dados em repouso e em movimento por padrão.
- Monitora o servidor e fornece recursos para insights de desempenho de consulta e recomendações de desempenho.

Com o Banco de Dados do Azure para PostgreSQL, você pode continuar a administrar o seu banco de dados. Mas você não precisa mais gerenciar o mecanismo de banco de dados, o sistema operacional ou o hardware. Exemplos de itens que você pode continuar a administrar incluem:

- Bancos de dados
- Conexão
- Ajuste do índice
- Ajuste de consulta
- Auditoria
- Segurança

Além disso, a configuração da alta disponibilidade para outro data center requer pouca ou nenhuma

administração e configuração.

- Com o PostgreSQL nas VMs do Azure, você tem controle total sobre o sistema operacional e a configuração da instância do servidor PostgreSQL. Com uma VM, você decide quando atualizar ou fazer upgrade do sistema operacional e do software de banco de dados e quais patches aplicar. Você também decide quando instalar qualquer software adicional, como um aplicativo antivírus. Alguns recursos automatizados são fornecidos para simplificar muito a aplicação de patches, backup e alta disponibilidade. Você pode controlar o tamanho da VM, o número de discos e as configurações de armazenamento desses discos. Para obter mais informações, confira [Tamanhos da máquina virtual e do serviço de nuvem para o Azure](#).

Tempo para mover para o Serviço PostgreSQL do Azure (PaaS)

- O Banco de Dados do Azure para PostgreSQL é a solução adequada para aplicativos projetados em nuvem quando a produtividade do desenvolvedor e um tempo de entrega rápido para novas soluções são fatores críticos. Com a funcionalidade de programação como DBA, o serviço é adequado para desenvolvedores e arquitetos de nuvem, pois reduz a necessidade de gerenciamento do sistema operacional e do banco de dados subjacentes.
- Quando você quiser economizar tempo e as despesas de adquirir um novo hardware local, o PostgreSQL em VMs do Azure será a solução adequada para aplicativos que exigem um controle granular e cuja personalização do mecanismo PostgreSQL não tem suporte no serviço ou exige acesso ao sistema operacional subjacente.

Próximas etapas

- Confira o [Preço do Banco de Dados do Azure para PostgreSQL](#).
- Comece com a criação do seu primeiro servidor.

Noções básicas sobre as mudanças na alteração da AC raiz para o servidor único do banco de Dados do Azure para PostgreSQL

21/05/2021 • 8 minutes to read

O servidor único do banco de Dados do Azure para PostgreSQL concluiu com êxito a alteração do certificado raiz em **15 de fevereiro de 2021 (15/02/2021)** como parte das práticas recomendadas de manutenção e segurança padrão. Este artigo fornece mais detalhes sobre as alterações, os recursos afetados e as etapas necessárias para garantir que o aplicativo mantenha a conectividade com o servidor do banco de dados.

Por que a atualização do certificado raiz é necessária?

Os usuários do banco de dados do Azure para PostgreSQL só podem usar o certificado predefinido para se conectar ao servidor PostgreSQL, que está localizado [aqui](#). No entanto, o [fórum do Navegador da AC \(autoridade de certificação\)](#) publicou recentemente relatórios de vários certificados emitidos por fornecedores de AC que não estavam em conformidade.

De acordo com os requisitos de conformidade do setor, os fornecedores de AC começaram a revogar certificados de AC para ACs não compatíveis, exigindo que os servidores usem certificados emitidos por ACs em conformidade e assinados por certificados de autoridade de certificação dessas ACs em conformidade. Como o Banco de Dados do Azure para MySQL usava um desses certificados não compatíveis, precisávamos alternar o certificado para a versão em conformidade para minimizar a possível ameaça aos servidores MySQL.

O novo certificado já foi implementado e estará em vigor a partir de 15 de fevereiro de 2021 (15/02/2021).

Qual alteração foi realizada em 15 de fevereiro de 2021 (15/02/2021)?

Em 15 de fevereiro de 2021, o [certificado raiz BaltimoreCyberTrustRoot](#) foi substituído por uma [versão em conformidade](#) do mesmo [certificado raiz BaltimoreCyberTrustRoot](#) para garantir que os clientes existentes não precisem alterar nada e não haja nenhum impacto nas conexões com o servidor. Durante essa alteração, o [certificado raiz BaltimoreCyberTrustRoot](#) não foi substituído por [DigiCertGlobalRootG2](#) e essa alteração foi adiada para permitir mais tempo para que os clientes façam a alteração.

É preciso fazer alterações no cliente para manter a conectividade?

Não é preciso fazer nenhuma alteração no lado do cliente. Se você seguiu nossa recomendação anterior abaixo, ainda poderá se conectar, desde que o [certificado BaltimoreCyberTrustRoot](#) não seja removido do certificado da AC combinado. É [recomendável não remover o BaltimoreCyberTrustRoot do certificado de AC combinada, até que seja avisado o contrário, para manter a conectividade](#).

Recomendação anterior

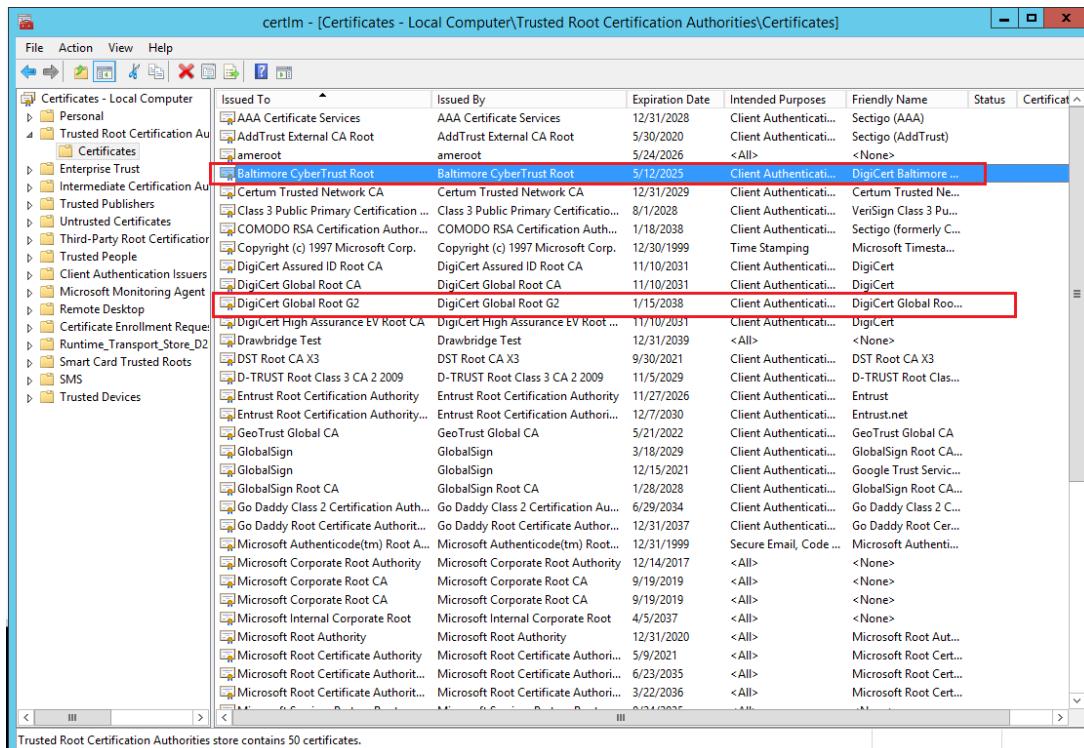
- Baixe o AC raiz BaltimoreCyberTrustRoot & DigiCertGlobalRootG2 dos seguintes links:
 - <https://www.digicert.com/CA Certs/BaltimoreCyberTrustRoot.crt.pem>
 - <https://cacerts.digicert.com/DigiCertGlobalRootG2.crt.pem>
- Gere um repositório de certificados de AC combinada com os certificados [BaltimoreCyberTrustRoot](#) e [DigiCertGlobalRootG2](#) incluídos.
 - Para usuários do Java (PostgreSQL JDBC) usando DefaultJavaSSLFactory, execute:

```
keytool -importcert -alias PostgreSQLServerCACert -file D:\BaltimoreCyberTrustRoot.crt.pem -keystore truststore -storepass password -noprompt
```

```
keytool -importcert -alias PostgreSQLServerCACert2 -file D:\DigiCertGlobalRootG2.crt.pem -keystore truststore -storepass password -noprompt
```

Depois, substitua o arquivo de repositório de chaves original pelo novo gerado:

- `System.setProperty("javax.net.ssl.trustStore","path_to_truststore_file");`
- `System.setProperty("javax.net.ssl.trustStorePassword","password");`
- Para usuários do .NET (Npgsql) no Windows, certifique-se de que **Baltimore Cybertrust Root G2 e DigiCert Global Root G2** existam no repositório de certificados do Windows, autoridades de certificação raiz confiáveis. Se não encontrar algum deles, importe o certificado ausente.



- Para usuários do .NET (Npgsql) no Linux usando `SSL_CERT_DIR`, verifique se **BaltimoreCyberTrustRoot** e **DigiCertGlobalRootG2** existem no diretório indicado por `SSL_CERT_DIR`. Se não encontrar algum deles, crie o arquivo do certificado ausente.
- Para outros usuários cliente do PostgreSQL, você pode mesclar dois arquivos de certificado de autoridade de certificação como este formato abaixo

-----INICIAR CERTIFICADO-----

(Root CA1: BaltimoreCyberTrustRoot.crt.pem)

-----CONCLUIR CERTIFICADO-----

-----INICIAR CERTIFICADO-----

(Root CA2: DigiCertGlobalRootG2.crt.pem)

-----CONCLUIR CERTIFICADO-----

- Substitua o arquivo pem da AC raiz original pelo arquivo de AC raiz combinada e reinicie o aplicativo/cliente.
- Futuramente, após o novo certificado ser implantado no lado do servidor, você poderá alterar o arquivo pem da AC para DigiCertGlobalRootG2.crt.pem.

NOTE

Não remova nem altere o certificado **Baltimore** até que a alteração de certificado seja feita. Enviaremos uma comunicação depois que a alteração for feita, após a qual será seguro remover o certificado **Baltimore**.

Por que o certificado **BaltimoreCyberTrustRoot** não foi substituído por **DigiCertGlobalRootG2** durante essa alteração em 15 de fevereiro de 2021?

Avaliamos a preparação do cliente para essa alteração e percebemos que muitos clientes precisavam de um prazo de entrega maior para gerenciar essa alteração. Para fins de fornecer um prazo de entrega maior aos clientes para preparação, decidimos adiar a alteração do certificado para **DigiCertGlobalRootG2** por pelo menos um ano, para um prazo de entrega suficiente aos clientes e usuários finais.

Recomendamos que os usuários usem as etapas mencionadas anteriormente para criar um certificado combinado e se conectar ao servidor, mas sem remover o certificado **BaltimoreCyberTrustRoot** até enviarmos uma comunicação para removê-lo.

O que acontece se removermos o certificado **BaltimoreCyberTrustRoot**?

Você começará a ter erros de conectividade ao conectar-se ao servidor do Banco de Dados do Azure para PostgreSQL. Você precisará configurar o SSL com certificado **BaltimoreCyberTrustRoot** novamente para manter a conectividade.

Perguntas frequentes

1. Se eu não estiver usando SSL/TLS, preciso atualizar a AC raiz mesmo assim?

Nenhuma ação é necessária se você não estiver usando SSL/TLS.

2. Se eu estiver usando SSL/TLS, preciso reiniciar meu servidor de banco de dados para atualizar a AC raiz?

Não, você não precisa reiniciar o servidor de banco de dados para começar a usar o novo certificado. Esta é uma alteração no lado do cliente, e as conexões de entrada do cliente precisam usar o novo certificado para que possam se conectar ao servidor de banco de dados.

3. Como fazer para saber se estou usando SSL/TLS com verificação de certificado raiz?

Você pode identificar se suas conexões verificam o certificado raiz ao revisar a cadeia de conexão.

- Se a cadeia de conexão incluir `sslmode=verify-ca` ou `sslmode=verify-full`, você precisará atualizar o certificado.
- Se a cadeia de conexão incluir `sslmode=disable`, `sslmode=allow`, `sslmode=prefer` ou `sslmode=require`, você não precisará atualizar os certificados.
- Se a cadeia de conexão não especificar `sslmode`, você não precisará atualizar os certificados.

Se estiver usando um cliente que abstrai a cadeia de conexão, revise a documentação do cliente para entender se ele verifica os certificados. Para entender o PostgreSQL `sslmode`, consulte as [descrições do modo SSL](#) na documentação do PostgreSQL.

4. Qual o impacto se estiver usando o Serviço de Aplicativo com o Banco de Dados do Azure para PostgreSQL?

Para os serviços de aplicativo do Azure que se conectam ao Banco de Dados do Azure para PostgreSQL, há dois cenários possíveis e dependem de como você está usando SSL com o aplicativo.

- Este novo certificado foi adicionado ao Serviço de Aplicativo no nível da plataforma. Se estiver usando os certificados SSL incluídos na plataforma do Serviço de Aplicativo no aplicativo, nenhuma ação será necessária.
- Se você estiver incluindo explicitamente o caminho para o arquivo de certificado SSL em seu código, precisará baixar o novo certificado e atualizar o código para usá-lo. Um bom exemplo desse cenário é quando você usa contêineres personalizados no serviço de aplicativo, como compartilhado na [Documentação do Serviço de Aplicativo](#)

5. Qual o impacto se estiver usando o AKS (Serviço de Kubernetes do Azure) com o Banco de Dados do Azure para PostgreSQL?

Se estiver tentando se conectar ao Banco de Dados do Azure para PostgreSQL usando os AKS (Serviço de Kubernetes do Azure), o acesso será feito de forma semelhante a partir de um ambiente de host de clientes dedicado. Consulte as etapas [aqui](#).

6. Qual o impacto se estiver usando o Azure Data Factory para me conectar ao Banco de Dados do Azure para PostgreSQL?

Para um conector que usa o Azure Integration Runtime, o conector aproveita os certificados no Repositório de Certificados do Windows no ambiente hospedado do Azure. Esses certificados já são compatíveis com os certificados aplicados recentemente, o que não exige nenhuma ação.

Para um conector que usa o Integration Runtime auto-hospedado em que você inclui explicitamente o caminho para o arquivo de certificado SSL na cadeia de conexão, será preciso baixar o [novo certificado](#) e atualizar a cadeia de conexão para usá-lo.

7. Preciso planejar um tempo de inatividade de manutenção do servidor de banco de dados para essa alteração?

Não. Como a alteração aqui é feita apenas no lado do cliente para se conectar ao servidor de banco de dados, não é necessário um tempo de inatividade de manutenção para o servidor de banco de dados para essa alteração.

8. Se criar um novo servidor após 15 de fevereiro de 2021 (15/02/2021), serei afetado?

Para servidores criados após 15 de fevereiro de 2021 (15/02/2021), você continuará a usar o [BaltimoreCyberTrustRoot](#) para a conexão dos aplicativos usando SSL.

9. Com que frequência a Microsoft atualiza os certificados ou qual é a política de expiração?

Esses certificados usados pelo Banco de Dados do Azure para PostgreSQL são fornecidos por ACs (autoridades de certificação) confiáveis. Portanto, o suporte desses certificados está vinculado ao suporte desses certificados pela AC. O certificado [BaltimoreCyberTrustRoot](#) está agendado para expirar em 2025, portanto, a Microsoft precisará realizar uma alteração de certificado antes da expiração. Em caso de bugs imprevistos nesses certificados predefinidos, a Microsoft precisará fazer a rotação do certificado o mais cedo possível, semelhante à alteração realizada em 15 de fevereiro de 2021, para garantir que o serviço esteja seguro e em conformidade em todos os momentos.

10. Se eu estiver usando réplicas de leitura, preciso fazer essa atualização somente no servidor primário ou em todas as réplicas de leitura?

Como essa atualização é uma alteração feita no lado do cliente, se o cliente liga os dados do servidor de réplica, você também precisará aplicar as alterações para esses clientes.

11. Há uma consulta do lado do servidor para verificar se o SSL está sendo usado?

Para verificar se você está usando a conexão SSL para se conectar ao servidor, confira [Verificação de SSL](#).

12. Preciso fazer alguma coisa se eu já tiver o DigiCertGlobalRootG2 no meu arquivo de certificado?

Não. Não é preciso fazer nada se o arquivo de certificado já tiver o DigiCertGlobalRootG2.

13. E se você estiver usando a imagem do Docker do PgBouncer sidecar fornecido pela Microsoft?

Uma nova imagem do Docker que dá suporte a [Baltimore e DigiCert](#) é publicada abaixo [aqui](#) (marca mais recente). Você pode extrair essa nova imagem para evitar qualquer interrupção na conectividade a partir de 15 de fevereiro de 2021.

14. E se eu tiver outras dúvidas?

Em caso de dúvidas, fale com os especialistas da comunidade no [Microsoft Q&A](#). Se tiver um plano de suporte e precisar de ajuda técnica, [entre em contato conosco](#)

Início Rápido: Usar o GitHub Actions para se conectar ao PostgreSQL do Azure

21/05/2021 • 4 minutes to read

APLICA-SE A: Banco de Dados do Azure para PostgreSQL – Servidor Único Banco de Dados do Azure para PostgreSQL – Servidor Flexível

Comece a usar o [GitHub Actions](#) usando um fluxo de trabalho para implantar atualizações de banco de dados no [Banco de Dados do Azure para PostgreSQL](#).

Pré-requisitos

Serão necessários:

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Um repositório do GitHub com os dados de exemplo (`data.sql`). Se você não tiver uma conta do GitHub, [inscreva-se gratuitamente](#).
- Um Banco de Dados do Azure para servidor PostgreSQL.
 - [Início Rápido: Criar um Banco de Dados do Azure para o servidor PostgreSQL no portal do Azure](#)

Visão geral do arquivo do fluxo de trabalho

Um fluxo de trabalho do GitHub Actions é definido por um arquivo YAML (.yml) no caminho `/.github/workflows/` no repositório. Essa definição contém as várias etapas e os parâmetros que compõem o fluxo de trabalho.

O arquivo tem duas seções:

SEÇÃO	TAREFAS
Autenticação	1. Definir uma entidade de serviço. 2. Criar um segredo do GitHub.
Implantar	1. Implantar o banco de dados.

Gerar as credenciais de implantação

Crie uma [entidade de serviço](#) com o comando `az ad sp create-for-rbac` na [CLI do Azure](#). Execute esse comando com o [Azure Cloud Shell](#) no portal do Azure ou selecionando o botão **Experimentar**.

Substitua os espaços reservados `{server-name}` pelo nome do servidor PostgreSQL hospedado no Azure.

Substitua `{subscription-id}` e `{resource-group}` pela ID da assinatura e pelo grupo de recursos conectados ao servidor PostgreSQL.

```
az ad sp create-for-rbac --name {server-name} --role contributor \
    --scopes /subscriptions/{subscription-id}/resourceGroups/{resource-group} \
    --sdk-auth
```

A saída é um objeto JSON com as credenciais de atribuição de função que fornecem acesso ao banco de dados

semelhante abaixo. Copie esse objeto JSON de saída para mais tarde.

```
{  
    "clientId": "<GUID>",  
    "clientSecret": "<GUID>",  
    "subscriptionId": "<GUID>",  
    "tenantId": "<GUID>",  
    (...)  
}
```

IMPORTANT

É sempre uma boa prática permitir acesso mínimo. O escopo no exemplo anterior é limitado ao servidor específico e não ao grupo de recursos inteiro.

Copiar a cadeia de conexão do PostgreSQL

No portal do Azure, acesse o Banco de Dados do Azure para servidor PostgreSQL e abra **Configurações > Cadeias de conexão**. Copie a cadeia de conexão ADO.NET. Substitua os valores de espaço reservado por `your_database` e `your_password`. A cadeia de conexão será semelhante ao exemplo a seguir.

IMPORTANT

- Para o Servidor único, use `user=adminusername@servername`. Observe que `@servername` é obrigatório.
- Para o Servidor flexível, use `user= adminusername` sem o `@servername`.

```
psql host={servername.postgres.database.azure.com} port=5432 dbname={your_database} user={adminusername}  
password={your_database_password} sslmode=require
```

Você usará a cadeia de conexão como um segredo do GitHub.

Configurar os segredos do GitHub

- Em [GitHub](#), procure seu repositório.
- Selecione **Configurações > Segredos > Novo segredo**.
- Cole toda a saída JSON do comando da CLI do Azure no campo valor do segredo. Dê ao segredo o nome `AZURE_CREDENTIALS`.

Ao configurar o arquivo de fluxo de trabalho posteriormente, você usa o segredo para o `creds` de entrada da ação de Logon do Azure. Por exemplo:

```
- uses: azure/login@v1  
  with:  
    creds: ${{ secrets.AZURE_CREDENTIALS }}
```

- Selecione **Novo segredo** novamente.
- Cole o valor da cadeia de conexão no campo de valor do segredo. Dê ao segredo o nome `AZURE_POSTGRES_CONNECTION_STRING`.

Adicionar seu fluxo de trabalho

1. Acesse **Ações** para seu repositório do GitHub.
2. Selecione **Configurar seu fluxo de trabalho por conta própria**.
3. Exclua tudo depois da seção `on:` do seu arquivo de fluxo de trabalho. Por exemplo, o fluxo de trabalho restante pode ter a aparência a seguir.

```
name: CI

on:
push:
  branches: [ master ]
pull_request:
  branches: [ master ]
```

4. Renomeie o fluxo de trabalho `PostgreSQL for GitHub Actions` e adicione as ações de check-out e logon. Essas ações farão o check-out do código do site e a autenticação com o Azure usando o segredo do GitHub `AZURE_CREDENTIALS` criado anteriormente.

```
name: PostgreSQL for GitHub Actions

on:
push:
  branches: [ master ]
pull_request:
  branches: [ master ]

jobs:
build:
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v1
    - uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}
```

5. Use a ação **Implantar do PostgreSQL** do Azure para se conectar à instância do PostgreSQL. Substitua `POSTGRESQL_SERVER_NAME` pelo nome do seu servidor. Você deverá ter um arquivo de dados do PostgreSQL chamado `data.sql` no nível raiz do repositório.

```
- uses: azure/postgresql@v1
  with:
    connection-string: ${{secrets.AZURE_POSTGRESQL_CONNECTION_STRING}}
    server-name: POSTGRESQL_SERVER_NAME
    sql-file: './data.sql'
```

6. Conclua o fluxo de trabalho adicionando uma ação para fazer logoff do Azure. Este é o fluxo de trabalho concluído. O arquivo será exibido na pasta `.github/workflows` do seu repositório.

```

name: PostgreSQL for GitHub Actions

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v1
      - uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

      - uses: azure/postgresql@v1
      with:
        server-name: POSTGRESQL_SERVER_NAME
        connection-string: ${{secrets.AZURE_POSTGRESQL_CONNECTION_STRING}}
        sql-file: './data.sql'

        # Azure logout
      - name: logout
        run: |
          az logout

```

Examinar sua implantação

1. Acesse **Ações** para seu repositório do GitHub.
2. Abra o primeiro resultado para ver os logs detalhados da execução do fluxo de trabalho.

Create main.yml .github/workflows/main.yml #1

Step	Duration
Set up job	18s
Run actions/checkout@v2.3.2	1s
Run Azure/login@v1	15s
Run azure/postgresql@v1	1s
Post Run actions/checkout@v2.3.2	0s
Complete job	0s

Limpar os recursos

Quando o banco de dados PostgreSQL do Azure e o repositório não forem mais necessários, limpe os recursos implantados excluindo o grupo de recursos e o repositório GitHub.

Próximas etapas

[Saiba mais sobre a integração do Azure com o GitHub](#)

[Saiba como se conectar ao servidor](#)

Migrar seu banco de dados PostgreSQL usando despejar e restaurar

21/05/2021 • 4 minutes to read

APLICA-SE A: Banco de Dados do Azure para PostgreSQL – Servidor Único Banco de Dados do Azure para PostgreSQL – Servidor Flexível

Use [pg_dump](#) para extrair um banco de dados PostgreSQL para um arquivo de despejo. Em seguida, use [pg_restore](#) para restaurar o banco de dados PostgreSQL de um arquivo morto criado pelo [pg_dump](#).

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Um [Servidor de Banco de Dados do Azure para PostgreSQL](#) com regras de firewall para a permissão do acesso.
- Utilitários de linha de comando [pg_dump](#) e [pg_restore](#) instalados.

Criar um arquivo de despejo que contenha os dados a serem carregados

Para fazer backup de um banco de dados PostgreSQL que existe localmente ou em uma VM, execute o seguinte comando:

```
pg_dump -Fc -v --host=<host> --username=<name> --dbname=<database name> -f <database>.dump
```

Por exemplo, se você tiver um servidor local e, nele, um banco de dados chamado **testdb**:

```
pg_dump -Fc -v --host=localhost --username=masterlogin --dbname=testdb -f testdb.dump
```

Restaurar os dados no banco de dados de destino

Depois de criar o banco de dados de destino, você poderá usar o comando [pg_restore](#) e o parâmetro [--dbname](#) para restaurar os dados no banco de dados de destino a partir do arquivo de despejo.

```
pg_restore -v --no-owner --host=<server name> --port=<port> --username=<user-name> --dbname=<target database name> <database>.dump
```

A inclusão do parâmetro [--no-owner](#) faz com que todos os objetos criados durante a restauração sejam de propriedade do usuário especificado com [--username](#). Para obter mais informações, consulte a [Documentação do PostgreSQL](#).

NOTE

Em servidores do Banco de Dados do Azure para PostgreSQL, as conexões TLS/SSL estão ativas por padrão. Se o seu servidor PostgreSQL exigir conexões TLS/SSL e não as tiver, defina uma variável de ambiente `PGSSLMODE=require` para que a ferramenta de `pg_restore` se conecte ao TLS. Sem TLS, pode ser exibido o seguinte erro: "FATAL: requer uma conexão SSL. Especifique as opções de SSL e tente novamente". Na linha de comando do Windows, execute o comando `SET PGSSLMODE=require` antes de executar o comando `pg_restore`. No Linux ou no Bash, execute o comando `export PGSSLMODE=require` antes de executar o comando `pg_restore`.

Neste exemplo, restauramos os dados no arquivo de despejo `testdb.dump` no banco de dados `mypgsql` no servidor de destino `mydemoserver.postgres.database.azure.com`.

Aqui está um exemplo de como usar o `pg_restore` em servidor único:

```
pg_restore -v --no-owner --host=mydemoserver.postgres.database.azure.com --port=5432 --username=mylogin@mydemoserver --dbname=mypgsql testdb.dump
```

Aqui está um exemplo de como usar o `pg_restore` em servidor flexível:

```
pg_restore -v --no-owner --host=mydemoserver.postgres.database.azure.com --port=5432 --username=mylogin --dbname=mypgsql testdb.dump
```

Otimizar o processo de migração

Uma maneira de migrar seu banco de dados PostgreSQL existente para o serviço Banco de Dados do Azure para PostgreSQL é fazer o backup do banco de dados na origem e restaurá-lo no Azure. Para minimizar o tempo necessário para concluir a migração, considere usar os seguintes parâmetros com os comandos de backup e restauração.

NOTE

Para obter informações de sintaxe detalhadas, confira os artigos [pg_dump](#) e [pg_restore](#).

Para o backup

Faça o backup com a opção `-Fc` para que você possa executar a restauração em paralelo e assim acelerá-la. Por exemplo:

```
pg_dump -h my-source-server-name -U source-server-username -Fc -d source-database-name -f Z:\Data\Backups\my-database-backup.dump
```

Para a restauração

- Mova o arquivo de backup para uma máquina virtual do Azure na mesma região do servidor do Banco de Dados do Azure para PostgreSQL servidor para o qual você está migrando. Execute o `pg_restore` nessa máquina virtual para reduzir a latência de rede. Criar a sua máquina virtual com [rede acelerada](#) habilitada.
- Abra o arquivo de despejo para verificar se as instruções `create index` aparecem após a inserção dos dados. Se não estiverem, coloque as instruções `create index` após a inserção dos dados. Elas costumam já estar assim por padrão, mas é bom confirmar.
- Restaure usando as opções `-Fc` e `-j` (com um número) para parallelizar a restauração. O número

especificado é o número de núcleos no servidor de destino. Você também pode experimentar com o dobro do número de núcleos no servidor de destino para testar o impacto.

Aqui está um exemplo de como usar o `pg_restore` em servidor único:

```
pg_restore -h my-target-server.postgres.database.azure.com -U azure-postgres-username@my-target-server -Fc -j 4 -d my-target-databasename Z:\Data\Backups\my-database-backup.dump
```

Aqui está um exemplo de como usar o `pg_restore` em servidor flexível:

```
pg_restore -h my-target-server.postgres.database.azure.com -U azure-postgres-username@my-target-server -Fc -j 4 -d my-target-databasename Z:\Data\Backups\my-database-backup.dump
```

- Você também pode editar o arquivo de despejo adicionando o comando `set synchronous_commit = off;` no início e o comando `set synchronous_commit = on;` no final. Se ele não for ativado no final, antes que os aplicativos alterem os dados, poderá ocorrer uma perda de dados.
- No servidor de destino do Banco de Dados do Azure para PostgreSQL, considere o seguinte antes da restauração:
 - Desative o rastreamento de desempenho de consulta. Essas estatísticas não são necessárias durante a migração. Você pode fazer isso definindo `pg_stat_statements.track`, `pg_qs.query_capture_mode` e `pgms_wait_sampling.query_capture_mode` para `NONE`.
 - Use uma SKU de alta computação e memória alta, como 32 e otimização de memória vCore, para acelerar a migração. Depois de concluir a restauração, você pode facilmente redimensionar para a SKU de sua preferência. Quanto maior a SKU, maior o paralelismo alcançado aumentando o parâmetro `-j` correspondente no comando `pg_restore`.
 - A presença de mais IOPS no servidor de destino pode melhorar o desempenho da restauração. Você pode provisionar mais IOPS aumentando o tamanho de armazenamento do servidor. Essa configuração não é reversível, mas avalie se um IOPS mais alto pode beneficiar a sua carga de trabalho real no futuro.

Lembre-se de testar e validar esses comandos em um ambiente de teste antes de usá-los em produção.

Próximas etapas

- Para migrar o seu banco de dados PostgreSQL usando exportar e importar, veja [Migrar um banco de dados PostgreSQL usando exportar e importar](#).
- Para obter mais informações de como migrar bancos de dados para o Banco de Dados do Azure para PostgreSQL, confira o [Guia de Migração de Banco de Dados](#).

Migrar seu banco de dados PostgreSQL usando exportar e importar

21/05/2021 • 2 minutes to read

APLICA-SE A: Banco de Dados do Azure para PostgreSQL – Servidor Único Banco de Dados do Azure para PostgreSQL – Servidor Flexível

Use [pg_dump](#) para extrair um banco de dados PostgreSQL para um arquivo de script, e [psql](#) para importar os dados para o banco de dados de destino desse arquivo.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Um [Servidor de Banco de Dados do Azure para PostgreSQL](#) com regras de firewall a fim de permitir o acesso e um banco de dados sob ele.
- O utilitário de linha de comando [pg_dump](#) instalado
- O utilitário de linha de comando [psql](#) instalado

Execute estas etapas para exportar e importar o banco de dados PostgreSQL.

Criar um arquivo de script usando pg_dump que contém os dados a serem carregados

Para exportar seu banco de dados PostgreSQL existente localmente ou em uma VM para um arquivo de script sql, execute o seguinte comando em seu ambiente existente:

```
pg_dump --host=<host> --username=<name> --dbname=<database name> --file=<database>.sql
```

Por exemplo, se você tiver um servidor local e um banco de dados chamado **testdb** nele:

```
pg_dump --host=localhost --username=masterlogin --dbname=testdb --file=testdb.sql
```

Importar os dados no Banco de Dados do Azure para PostgreSQL de destino

Você pode usar a linha de comando psql e o parâmetro --dbname (-d) para importar os dados para o servidor do Banco de Dados do Azure para PostgreSQL e carregar o arquivo sql.

```
psql --file=<database>.sql --host=<server name> --port=5432 --username=<user> --dbname=<target database name>
```

Este exemplo usa um utilitário psql e um arquivo de script nomeado **testdb.sql** da etapa anterior para importar dados para o banco de dados **mypgsqldb** no servidor de destino **mydemoserver.postgres.database.azure.com**.

Para um **Servidor individual**, use este comando

```
psql --file=testdb.sql --host=mydemoserver.database.windows.net --port=5432 --username=mylogin@mydemoserver  
--dbname=mypgsql
```

Para **Servidor flexível**, use este comando

```
psql --file=testdb.sql --host=mydemoserver.database.windows.net --port=5432 --username=mylogin --  
dbname=mypgsql
```

Próximas etapas

- Para migrar um banco de dados PostgreSQL usando despejar e restaurar, veja [Migrar seu banco de dados PostgreSQL usando despejar e restaurar](#).
- Para obter mais informações de como migrar bancos de dados para o Banco de Dados do Azure para PostgreSQL, confira o [Guia de Migração de Banco de Dados](#).

Migração com tempo de inatividade mínimo para servidor único do Banco de Dados do Azure para PostgreSQL

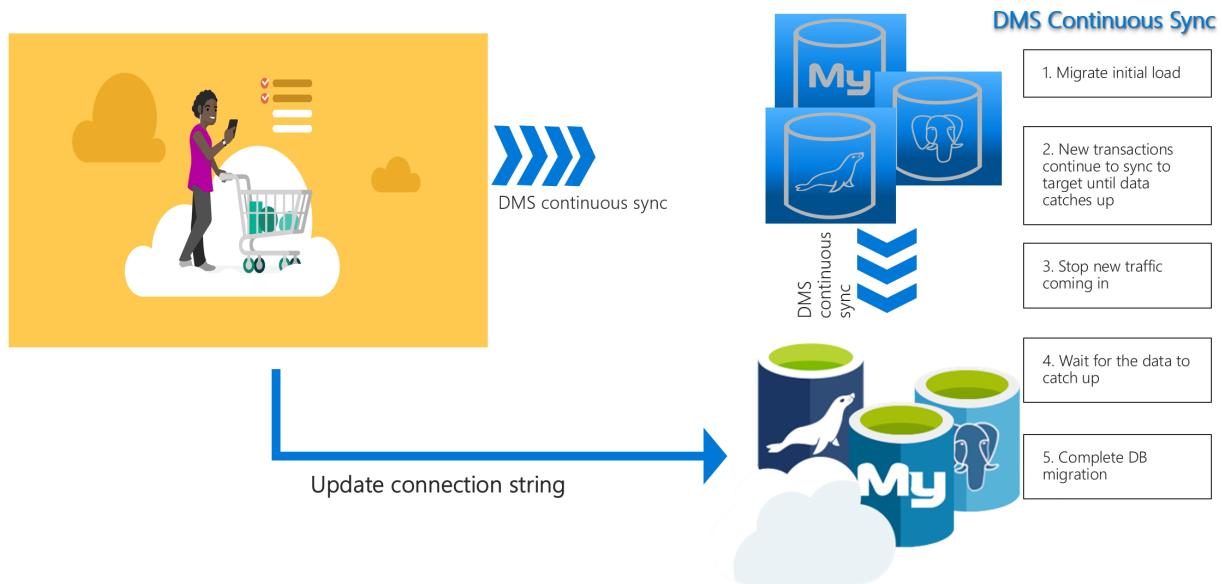
21/05/2021 • 2 minutes to read

APLICA-SE A: ✓ Banco de Dados do Azure para PostgreSQL – ✓ Servidor Único Banco de Dados do Azure para PostgreSQL – Servidor Flexível ✓ Banco de Dados do Azure para PostgreSQL - Hiperescala

É possível realizar migrações do PostgreSQL para o Banco de Dados do Azure para PostgreSQL com tempo de inatividade mínimo usando o **recurso de sincronização contínua** introduzido recentemente para [DMS](#) (Serviço de Migração de Banco de Dados do Azure). Essa funcionalidade limita o tempo de inatividade incorrido pelo aplicativo.

Visão geral

O DMS do Azure executa uma carga inicial do local no Banco de Dados do Azure para PostgreSQL e, em seguida, sincroniza continuamente quaisquer novas transações no Azure enquanto o aplicativo permanece em execução. Depois que os dados são atualizados no lado do Azure de destino, você interrompe o aplicativo por um breve momento (tempo de inatividade mínimo), aguarde o último lote de dados (desde o momento em que você para o aplicativo até que o aplicativo esteja efetivamente indisponível para receber qualquer novo tráfego) para atualizar no destino e, em seguida, atualize a cadeia de conexão para apontar para o Azure. Quando você terminar, o aplicativo estará ativo no Azure!



Próximas etapas

- Exibir o vídeo [Modernização de aplicativos com o Microsoft Azure](#), que contém uma demonstração mostrando como migrar aplicativos do PostgreSQL para o Banco de Dados do Azure para PostgreSQL.
- Consulte o tutorial [Migrar o PostgreSQL para o Banco de Dados do Azure para PostgreSQL on-line usando o DMS](#).

Migrar o Oracle para o Banco de Dados do Azure para PostgreSQL

09/08/2021 • 13 minutes to read

Este guia ajuda a migrar seu esquema do Oracle para o Banco de Dados do Azure para PostgreSQL.

Para obter diretrizes de migração detalhadas e abrangentes, confira os [Recursos do guia de migração](#).

Pré-requisitos

Para migrar seu esquema do Oracle para o Banco de Dados do Azure para PostgreSQL, você precisa:

- Verificar se o ambiente de origem tem suporte.
- Baixar a versão mais recente do [ora2pg](#).
- Ter a versão mais recente do [módulo DBD](#).

Visão geral

O PostgreSQL é um dos bancos de dados de software livre mais avançados do mundo. Este artigo descreve como usar a ferramenta ora2pg gratuita para migrar um banco de dados Oracle para PostgreSQL. Você pode usar o ora2pg para migrar um banco de dados Oracle ou MySQL para um esquema compatível com PostgreSQL.

A ferramenta ora2pg conecta seu banco de dados Oracle, faz sua verificação automaticamente e extrai sua estrutura ou seus dados. Em seguida, o ora2pg gera scripts SQL que você pode carregar em seu banco de dados PostgreSQL. Você pode usar ora2pg para tarefas como: engenharia reversa de um banco de dados Oracle, migração de um grande banco de dados corporativo ou simplesmente para replicar alguns dados Oracle em um banco de dados PostgreSQL. A ferramenta é fácil de usar e não requer nenhum conhecimento de banco de dados Oracle, além da capacidade de fornecer os parâmetros necessários para se conectar ao banco de dados.

NOTE

Para saber mais sobre como usar a versão mais recente do ora2pg, confira a [documentação do ora2pg](#).

Arquitetura de migração típica do ora2pg



Depois de provisionar a VM e o Banco de Dados do Azure para PostgreSQL, você precisará definir duas configurações para habilitar a conectividade entre elas: **Permitir o acesso aos serviços do Azure** e **Impor a conexão SSL**:

- Folha Conexão de Segurança > Permitir acesso aos serviços do Azure > ATIVADO

- Folha Conexão de Segurança > Configurações de SSL > Impor conexão SSL > DESATIVADO

Recomendações

- Para melhorar o desempenho das operações de avaliação ou de exportação no servidor Oracle, colete estatísticas:

```
BEGIN  
  
    DBMS_STATS.GATHER_SCHEMA_STATS  
    DBMS_STATS.GATHER_DATABASE_STATS  
    DBMS_STATS.GATHER_DICTIONARY_STATS  
END;
```

- Exporte dados usando o comando `COPY` em vez de `INSERT`.
- Evite exportar tabelas com suas FKs (chaves estrangeiras), restrições e índices. Esses elementos retardam o processo de importação de dados para o PostgreSQL.
- Crie exibições materializadas usando a *cláusula sem dados*. Em seguida, atualize as exibições mais tarde.
- Se possível, use índices exclusivos em exibições materializadas. Esses índices podem acelerar a atualização quando você usa a sintaxe `REFRESH MATERIALIZED VIEW CONCURRENTLY`.

Pré-migração

Depois de verificar se o seu ambiente de origem tem suporte e se você atente a todos os pré-requisitos, você está pronto para iniciar o estágio de pré-migração. Para começar:

1. **Descobrir:** inventariar os bancos de dados que você precisa migrar.
2. **Avaliar:** avalie esses bancos de dados para verificar se há possíveis problemas de migração ou bloqueadores.
3. **Converter:** resolver todos os itens descobertos.

Para migrações heterogêneas, como Oracle para Banco de Dados do Azure para PostgreSQL, esse estágio também envolve tornar os esquemas do banco de dados de origem compatíveis com o ambiente de destino.

Descobrir

A meta da fase de descoberta é identificar as fontes de dados existentes e os detalhes sobre os recursos que estão sendo usados. Essa fase ajuda você a entender e planejar melhor a migração. O processo envolve a verificação da rede para identificar todas as instâncias do Oracle da sua organização junto com a versão e os recursos em uso.

Scripts de pré-avaliação da Microsoft para Oracle são executados no banco de dados Oracle. Os scripts de pré-avaliação consultam os metadados do Oracle. Os scripts fornecem:

- Um inventário de banco de dados, incluindo contagens de objetos por esquema, tipo e status.
- Uma estimativa aproximada dos dados brutos em cada esquema, com base nas estatísticas.
- O tamanho das tabelas em cada esquema.
- O número de linhas de código por pacote, função, procedimento e assim por diante.

Baixe os scripts relacionados a partir do [GitHub](#).

Avaliar

Depois de inventariar os bancos de dados Oracle, você terá uma ideia do tamanho do banco e dos possíveis desafios. A próxima etapa é executar a avaliação.

Estimar o custo de uma migração do Oracle para o PostgreSQL não é fácil. Para avaliar o custo de migração, o

ora2pg verifica todos os objetos de banco de dados, funções e procedimentos armazenados para objetos e código PL/SQL que ele não pode converter automaticamente.

A ferramenta ora2pg tem um modo de análise de conteúdo que inspeciona o banco de dados Oracle para gerar um relatório de texto. O relatório descreve o que o banco de dados Oracle contém e o que não pode ser exportado.

Para ativar o modo de *análise e relatório*, use o tipo exportado, `SHOW_REPORT` conforme mostrado no seguinte comando:

```
ora2pg -t SHOW_REPORT
```

A ferramenta ora2pg pode converter o código SQL e PL/SQL da sintaxe do Oracle para PostgreSQL. Assim, depois que o banco de dados é analisado, o ora2pg pode estimar as dificuldades do código e o tempo necessário para migrar um banco de dados completo.

Para estimar o custo de migração em dias humanos, o ora2pg permite que você use uma diretiva de configuração chamada `ESTIMATE_COST`. Você também pode habilitar essa diretiva em um prompt de comando:

```
ora2pg -t SHOW_REPORT --estimate_cost
```

A unidade de migração padrão representa cerca de cinco minutos para um especialista em PostgreSQL. Se essa migração for a primeira, você poderá aumentar a unidade de migração padrão usando a diretiva de configuração `COST_UNIT_VALUE` ou a opção de linha de comando `--cost_unit_value`.

A última linha do relatório mostra o código de migração estimado total em dias humanos. A estimativa segue o número de unidades de migração estimadas para cada objeto.

No seguinte exemplo de código, você verá algumas variações de avaliação:

- Avaliação de tabelas
- Avaliação de colunas
- Avaliação de esquema que usa uma unidade de custo padrão de cinco minutos
- Avaliação de esquema que usa uma unidade de custo de dez minutos

```
ora2pg -t SHOW_TABLE -c c:\ora2pg\ora2pg_hr.conf > c:\ts303\hr_migration\reports\tables.txt  
ora2pg -t SHOW_COLUMN -c c:\ora2pg\ora2pg_hr.conf > c:\ts303\hr_migration\reports\columns.txt  
ora2pg -t SHOW_REPORT -c c:\ora2pg\ora2pg_hr.conf --dump_as_html --estimate_cost >  
c:\ts303\hr_migration\reports\report.html  
ora2pg -t SHOW_REPORT -c c:\ora2pg\ora2pg_hr.conf --cost_unit_value 10 --dump_as_html --estimate_cost >  
c:\ts303\hr_migration\reports\report2.html
```

Veja a saída do nível de migração de avaliação de esquema B-5:

- Níveis de migração:
 - A – migração que pode ser executada automaticamente
 - B – migração com reescrita de código e um custo de dias humanos de até cinco dias
 - C – migração com reescrita de código e um custo de dias humanos acima de cinco dias
- Níveis técnicos:
 - 1 = Trivial: nenhuma função armazenada e nenhum gatilho
 - 2 = Fácil: sem presença de função armazenada, mas de gatilhos; sem regravação manual

- 3 = Simples: funções e/ou gatilhos armazenados; sem regravação manual
- 4 = Manual: sem presença de função armazenada, mas de gatilhos ou exibições com regravação de código
- 5 = Difícil: funções armazenadas e/ou gatilhos com reescrita de código

A avaliação consiste em:

- Uma letra (A ou B) para especificar se a migração precisa de regravação manual.
- Um número de 1 a 5 para indicar a dificuldade técnica.

Outra opção, `-human_days_limit`, especifica o limite de dias humanos. Neste cenário, defina o nível de migração como C para indicar que a migração precisa de uma grande quantidade de trabalho, gerenciamento de projeto completo e suporte de migração. O padrão é de 10 dias humanos. Você pode usar a diretiva de configuração `HUMAN_DAYS_LIMIT` para alterar esse valor padrão permanentemente.

Essa avaliação de esquema foi desenvolvida para ajudar os usuários a decidir qual banco de dados deve ser migrado primeiro e quais equipes mobilizar.

Converter

Em migrações de tempo de inatividade mínimo, a origem da migração é alterada. Ela tem um descompasso em relação ao destino em termos de dados e esquema após a migração única. Durante a fase de *Sincronização de dados*, verifique se todas as alterações na origem são capturadas e aplicadas ao destino quase em tempo real. Depois de verificar se todas as alterações são aplicadas ao destino, você pode *migrar* do ambiente de origem para o ambiente de destino.

Nesta etapa da migração, o código Oracle e os scripts DDL são convertidos ou traduzidos para PostgreSQL. A ferramenta ora2pg exporta os objetos Oracle em um formato PostgreSQL automaticamente. Alguns dos objetos gerados não podem ser compilados no banco de dados PostgreSQL sem alterações manuais.

Para entender quais elementos precisam de intervenção manual, primeiro compile os arquivos gerados pelo ora2pg no banco de dados PostgreSQL. Verifique o log e faça as alterações necessárias até que a estrutura do esquema seja compatível com a sintaxe do PostgreSQL.

Criar um modelo de migração

Recomendamos o uso do modelo de migração que o ora2pg fornece. Quando você usa as opções `--project_base` e `--init_project`, o ora2pg cria um modelo de projeto com uma árvore de trabalho, um arquivo de configuração e um script para exportar todos os objetos do banco de dados Oracle. Para saber mais, confira a [documentação do ora2pg](#).

Use o seguinte comando:

```
ora2pg --project_base /app/migration/ --init_project test_project
```

Veja a saída de exemplo:

```
ora2pg --project_base /app/migration/ --init_project test_project
Creating project test_project.
/app/migration/test_project/
schema/
    dblinks/
    directories/
    functions/
    grants/
    mviews/
    packages/
    partitions/
    procedures/
    sequences/
    synonyms/
    tables/
    tablespaces/
    triggers/
    types/
    views/
sources/
    functions/
    mviews/
    packages/
    partitions/
    procedures/
    triggers/
    types/
    views/
data/
config/
reports/

Generating generic configuration file
Creating script export_schema.sh to automate all exports.
Creating script import_all.sh to automate all imports.
```

O diretório `sources/` contém o código Oracle. O diretório `schema/` contém o código portado para PostgreSQL. E o diretório `reports/` contém os relatórios HTML e a avaliação de custo de migração.

Depois que a estrutura do projeto é criada, um arquivo de configuração genérico é gerado. Definir a conexão de banco de dados Oracle e os parâmetros de configuração relevantes no arquivo de configuração. Para saber mais sobre o arquivo de configuração, confira a [documentação do ora2pg](#).

Exportar objetos Oracle

Em seguida, exporte os objetos Oracle como objetos PostgreSQL executando o arquivo `export_schema.sh`.

```
cd /app/migration/mig_project
./export_schema.sh
```

Execute o seguinte comando manualmente.

```
SET namespace="/app/migration/mig_project"

ora2pg -p -t DBLINK -o dblink.sql -b %namespace%/schema/dblinks -c %namespace%/config/ora2pg.conf
ora2pg -p -t DIRECTORY -o directory.sql -b %namespace%/schema/directories -c %namespace%/config/ora2pg.conf
ora2pg -p -t FUNCTION -o functions2.sql -b %namespace%/schema/functions -c %namespace%/config/ora2pg.conf
ora2pg -p -t GRANT -o grants.sql -b %namespace%/schema/grants -c %namespace%/config/ora2pg.conf
ora2pg -p -t MVIEW -o mview.sql -b %namespace%/schema/mviews -c %namespace%/config/ora2pg.conf
ora2pg -p -t PACKAGE -o packages.sql -b %namespace%/schema/packages -c %namespace%/config/ora2pg.conf
ora2pg -p -t PARTITION -o partitions.sql -b %namespace%/schema/partitions -c %namespace%/config/ora2pg.conf
ora2pg -p -t PROCEDURE -o procs.sql -b %namespace%/schema/procedures -c %namespace%/config/ora2pg.conf
ora2pg -p -t SEQUENCE -o sequences.sql -b %namespace%/schema/sequences -c %namespace%/config/ora2pg.conf
ora2pg -p -t SYNONYM -o synonym.sql -b %namespace%/schema/synonyms -c %namespace%/config/ora2pg.conf
ora2pg -p -t TABLE -o table.sql -b %namespace%/schema/tables -c %namespace%/config/ora2pg.conf
ora2pg -p -t TABLESPACE -o tablespaces.sql -b %namespace%/schema/tablespaces -c %namespace%/config/ora2pg.conf
ora2pg -p -t TRIGGER -o triggers.sql -b %namespace%/schema/triggers -c %namespace%/config/ora2pg.conf
ora2pg -p -t TYPE -o types.sql -b %namespace%/schema/types -c %namespace%/config/ora2pg.conf
ora2pg -p -t VIEW -o views.sql -b %namespace%/schema/views -c %namespace%/config/ora2pg.conf
```

Para extrair os dados, use o seguinte comando.

```
ora2pg -t COPY -o data.sql -b %namespace/data -c %namespace/config/ora2pg.conf
```

Compilar arquivos

Por fim, compile todos os arquivos no servidor do Banco de Dados do Azure para PostgreSQL. Você pode optar por carregar os arquivos DDL gerados manualmente ou usar o segundo script *import_all.sh*, para importar esses arquivos interativamente.

```
psql -f %namespace%\schema\sequences\sequence.sql -h server1-server.postgres.database.azure.com -p 5432 -U
username@server1-server -d database -l %namespace%\ schema\sequences\create_sequences.log

psql -f %namespace%\schema\tables\table.sql -h server1-server.postgres.database.azure.com p 5432 -U
username@server1-server -d database -l %namespace%\schema\tables\create_table.log
```

Veja o comando de importação de dados:

```
psql -f %namespace%\data\table1.sql -h server1-server.postgres.database.azure.com -p 5432 -U
username@server1-server -d database -l %namespace%\data\table1.log

psql -f %namespace%\data\table2.sql -h server1-server.postgres.database.azure.com -p 5432 -U
username@server1-server -d database -l %namespace%\data\table2.log
```

Enquanto os arquivos estão sendo compilados, verifique os logs e corrija qualquer sintaxe que o ora2pg não tenha conseguido converter.

Para saber mais, confira [Soluções alternativas de migração do Oracle para o Banco de Dados do Azure para PostgreSQL](#).

Migrar

Depois de atender os pré-requisitos necessários e concluir as etapas de pré-migração, você pode iniciar o esquema e a migração de dados.

Migrar esquema e dados

Depois de fazer as correções necessárias, uma compilação estável do banco de dados estará pronta para ser implantada. Execute os comandos de importação `psql`, apontando para os arquivos que contêm o código modificado. Esta tarefa compila os objetos do banco de dados no Banco de Dados PostgreSQL e importa os

dados.

Nesta etapa, você pode implementar um nível de paralelismo na importação dos dados.

Sincronizar dados e migrar

Nas migrações online (mínimo de tempo de inatividade), a origem de migração continua a mudar. Ela tem um descompasso em relação ao destino em termos de dados e esquema após a migração única.

Durante a fase de *Sincronização de dados*, verifique se todas as alterações na origem são capturadas e aplicadas ao destino quase em tempo real. Depois de verificar se todas as alterações foram aplicadas, você pode migrar do ambiente de origem para o de destino.

Para fazer uma migração online, entre em contato com o suporte de AskAzureDBforPostgreSQL@service.microsoft.com.

Em uma migração *delta/incremental* que usa ora2pg, para cada tabela, use uma consulta que filtra (*corta*) por data, hora ou outro parâmetro. Em seguida, conclua a migração usando uma segunda consulta que migre os dados restantes.

Na tabela de dados de origem, migre todos os dados históricos primeiro. Veja um exemplo:

```
select * from table1 where filter_data < 01/01/2019
```

Você pode consultar as alterações desde a migração inicial executando o seguinte comando:

```
select * from table1 where filter_data >= 01/01/2019
```

Nesse caso, recomendamos que você aprimore a validação verificando a paridade dos dados em ambos os lados, de origem e de destino.

Pós-migração

Após o estágio de *Migração*, conclua as tarefas de pós-migração para garantir que tudo esteja funcionando da maneira mais estável e eficiente possível.

Corrigir aplicativos

Depois que os dados são migrados para o ambiente de destino, todos os aplicativos que antes consumiam a origem, precisam começar a consumir o destino. Às vezes, a instalação requer alterações nos aplicativos.

Teste

Depois que os dados forem migrados para o destino, execute testes em relação aos bancos de dado para verificar se os aplicativos funcionam bem com o destino. Garanta que a origem e o destino sejam migrados corretamente executando os scripts de validação de dados manuais nos bancos de dados de origem Oracle e de destino PostgreSQL.

Idealmente, se os bancos de dados de origem e de destino tiverem um caminho de rede, o ora2pg deverá ser usado na validação de dados. Você pode executar a ação **TEST** para garantir que todos os objetos do banco de dados Oracle tenham sido criados no PostgreSQL.

Execute este comando:

```
ora2pg -t TEST -c config/ora2pg.conf > migration_diff.txt
```

Otimizar

A fase de pós-migração é crucial para reconciliar quaisquer problemas de precisão de dados e verificar a integridade. Nessa fase, você também resolve problemas de desempenho com a carga de trabalho.

Ativos de migração

Para saber mais sobre esse cenário de migração, confira os seguintes recursos. Eles dão suporte ao envolvimento do projeto de migração no mundo real.

RECURSO	DESCRIÇÃO
Manual de migração do Oracle para o Azure para PostgreSQL	Este documento ajuda arquitetos, consultores, administradores de banco de dados e funções relacionadas a migrar rapidamente cargas de trabalho do Oracle para o Banco de Dados do Azure para PostgreSQL usando o ora2pg.
Soluções alternativas de migração do Oracle para o Azure para PostgreSQL	Este documento ajuda arquitetos, consultores, administradores de banco de dados e funções relacionadas a corrigir ou solucionar problemas rapidamente durante a migração de cargas de trabalho do Oracle para o Banco de Dados do Azure para PostgreSQL.
Etapas para instalar o ora2pg no Windows ou Linux	Este documento fornece um guia de instalação rápida para migrar o esquema e os dados do Oracle para o Banco de Dados do Azure para PostgreSQL usando o ora2pg no Windows ou no Linux. Para saber mais, confira a documentação do ora2pg .

A equipe de engenharia de dados do SQL desenvolveu esses recursos. O objetivo principal desta equipe é desbloquear e acelerar a modernização complexa de projetos de migração de plataforma de dados para a plataforma de dados do Microsoft Azure.

Mais suporte

Para obter ajuda na migração além do escopo das ferramentas do ora2pg, entre em contato com [@Ask Banco de Dados do Azure para PostgreSQL](#).

Próximas etapas

Para obter uma matriz de serviços e ferramentas de migração de banco de dados e de data e tarefas especiais, confira [Serviços e ferramentas para a migração de dados](#).

Documentação:

- [Documentação do Banco de Dados do Azure para PostgreSQL](#)
- [Documentação do ora2pg](#)
- [Site do PostgreSQL](#)
- [Suporte a transações autônomas no PostgreSQL](#)

Política de versão do Banco de Dados do Azure para PostgreSQL

13/08/2021 • 4 minutes to read

Esta página descreve a política de versão do Banco de Dados do Azure para PostgreSQL e é aplicável ao Banco de Dados do Azure para PostgreSQL - Servidor único e Banco de Dados PostgreSQL do Azure - Modos de implantação do Servidor flexível (versão preliminar).

Versões do PostgreSQL com suporte

O Banco de Dados do Azure para PostgreSQL oferece suporte às seguintes versões de banco de dados.

VERSÃO	SERVIDOR ÚNICO	SERVIDOR FLEXÍVEL (VERSSÃO PRÉVIA)	HIPERESCALA (CITUS)
PostgreSQL 13		X	X*
PostgreSQL 12		X	X*
PostgreSQL 11	X	X	X
PostgreSQL 10	X		
PostgreSQL 9.6	X		
<i>PostgreSQL 9.5 (desativado)</i>	X		

(* O PostgreSQL 12 e 13 estão disponíveis como uma versão prévia do recurso no Hiperescala (Citus).)

Suporte da versão principal

Cada versão principal do PostgreSQL terá suporte do Banco de Dados do Azure para PostgreSQL a partir da data em que o Azure começa a dar suporte à versão até que a versão seja desativada pela Comunidade do PostgreSQL, conforme informado na [política de versão da comunidade PostgreSQL](#).

Suporte da versão secundária

O Banco de Dados do Azure para PostgreSQL executa automaticamente atualizações de versões secundárias para a versão do PostgreSQL preferida do Azure como parte da manutenção periódica.

Política de desativação da versão principal

A tabela a seguir fornece os detalhes de desativação para as versões principais do PostgreSQL. As datas seguem a [política de versão da comunidade do PostgreSQL](#).

VERSÃO	WHAT'S NEW	DATA DE INÍCIO DO SUPORTE DO AZURE	DATA DE BAIXA
PostgreSQL 9.5 (desativado)	Recursos	18 de abril de 2018	11 de fevereiro de 2021

VERSÃO	WHAT'S NEW	DATA DE INÍCIO DO SUPORTE DO AZURE	DATA DE BAIXA
PostgreSQL 9.6	Recursos	18 de abril de 2018	11 de novembro de 2021
PostgreSQL 10	Recursos	4 de junho de 2018	10 de novembro de 2022
PostgreSQL 11	Recursos	24 de julho de 2019	9 de novembro de 2023
PostgreSQL 12	Recursos	22 de setembro de 2020	14 de novembro de 2024
PostgreSQL 13	Recursos	25 de maio de 2021	13 de novembro de 2025

Versões do mecanismo PostgreSQL desativadas sem suporte no Banco de Dados do Azure para PostgreSQL

Você pode continuar a executar a versão desativada no Banco de Dados do Azure para PostgreSQL. No entanto, observe as seguintes restrições após a data de desativação para cada versão do banco de dados PostgreSQL:

- Como a comunidade não lançará correções de bugs ou correções de segurança adicionais, o Banco de Dados do Azure para PostgreSQL não corrigirá o mecanismo de banco de dados desativado em busca de bugs ou problemas de segurança e, de outra forma, não executará medidas de segurança em relação ao mecanismo de banco de dados desativado. Você pode enfrentar vulnerabilidades de segurança ou outros problemas como resultado. No entanto, o Azure continuará executando a manutenção periódica e aplicação de patch para o host, o sistema operacional, os contêineres e quaisquer outros componentes relacionados ao serviço.
- Se você enfrentar qualquer problema de suporte relacionado ao banco de dados PostgreSQL, talvez não seja possível obter suporte. Nesses casos, você precisará atualizar seu banco de dados para que possamos fornecer suporte.
- Você não poderá criar novos servidores de banco de dados para a versão desativada. No entanto, você poderá executar recuperações pontuais e criar réplicas de leitura para os servidores existentes.
- Novos recursos de serviço desenvolvidos pelo Banco de Dados do Azure para PostgreSQL só podem estar disponíveis para versões de servidor de banco de dados com suporte.
- Os SLAs de tempo de atividade serão aplicados exclusivamente a problemas relacionados ao serviço do Banco de Dados do Azure para PostgreSQL e não a qualquer tempo de inatividade causado por bugs relacionados ao mecanismo de banco de dados.
- No evento extremo de uma ameaça séria ao serviço causado pela vulnerabilidade do mecanismo do banco de dados PostgreSQL identificada na versão do banco de dados desativada, o Azure pode optar por interromper seu servidor de banco de dados para proteger o serviço. Nesse caso, você será notificado para atualizar o servidor antes de colocar o servidor online.

Sintaxe da versão do PostgreSQL

Antes do PostgreSQL versão 10, a [política de controle de versão do PostgreSQL](#) considerava uma atualização de *versão principal* como sendo um aumento no primeiro *ou* no segundo número. Por exemplo, 9.5 para 9.6 era considerado uma atualização de *versão principal*. Na versão 10 em diante, apenas uma alteração no primeiro número é considerada uma atualização de *versão principal*. Por exemplo, 10.0 para 10.1 é uma atualização de *versão secundária*. A versão 10 para 11 é uma atualização de *versão principal*.

Próximas etapas

- Consulte Banco de Dados do Azure para PostgreSQL – Servidor único [versões com suporte](#)
- Confira Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão preliminar) [versões com](#)

[suporte](#)

- Para obter informações sobre como executar atualizações de versão principais, consulte a documentação de [Atualizações da versão principal](#).
- Para obter informações sobre as extensões PostgreSQL compatíveis, confira [o documento de extensões](#).

Atualizar seu banco de dados PostgreSQL usando despejar e restaurar

09/08/2021 • 9 minutes to read

NOTE

Os conceitos explicados nesta documentação se aplicam ao Banco de Dados do Azure para PostgreSQL - Servidor Único e ao Banco de Dados do Azure para PostgreSQL - Servidor Flexível (versão prévia).

Você pode atualizar o servidor PostgreSQL implantado no Banco de Dados do Azure para PostgreSQL migrando seus bancos de dados para um servidor com uma versão principal mais atualizada usando os métodos a seguir.

- Método **offline** usando PostgreSQL [pg_dump](#) e [pg_restore](#) que incorre em tempo de inatividade para migração de dados. Este documento aborda esse método de atualização/migração.
- Método **online** usando o DMS ([Serviço de Migração de Banco de Dados](#)). Esse método fornece uma migração de tempo de inatividade reduzida, mantém o banco de dados de destino em sincronia com a origem e você pode escolher quando substituir. No entanto, há poucos pré-requisitos e restrições a serem abordados para o uso do DMS. Para obter detalhes, confira a [Documentação do DMS](#).

A tabela a seguir fornece algumas recomendações com base em tamanhos e cenários de banco de dados.

BANCO DE DADOS/CENÁRIO	DESPEJO/RESTAURAÇÃO (OFFLINE)	DMS (ONLINE)
Você tem um banco de dados pequeno e pode arcar com o tempo de inatividade para atualizar	X	
Bancos de dados pequenos (< 10 GB)	X	X
Bancos de pequenos-médios (10 GB a 100 GB)	X	X
Bancos de dados grandes (> 100 GB)		X
Pode arcar com o tempo de inatividade para atualizar (independentemente do tamanho do banco de dados)	X	
Pode atender aos pré-requisitos do DMS, incluindo uma reinicialização?		X
Pode evitar tabelas DDLs e não registradas durante o processo de atualização?		X

Este guia fornece algumas metodologias e exemplos de migração offline para mostrar como você pode migrar do servidor de origem para o servidor de destino que executa uma versão mais recente do PostgreSQL.

NOTE

O despejo e a restauração do PostgreSQL podem ser executados de várias maneiras. Você pode optar por migrar usando um dos métodos fornecidos neste guia ou escolher alguma forma alternativa para atender às suas necessidades. Para obter uma sintaxe detalhada de despejo e restauração com parâmetros adicionais, confira os artigos [pg_dump](#) e [pg_restore](#).

Pré-requisitos para usar despejo e restauração com o Banco de Dados PostgreSQL do Azure

Para seguir este guia de instruções, você precisa:

- Um servidor de banco de dados PostgreSQL de **origem** que executa uma versão inferior do mecanismo que você deseja atualizar.
- Um servidor de banco de dados PostgreSQL de **destino** com a versão principal desejada do servidor do [Banco de Dados do Azure para PostgreSQL - Servidor Único](#) ou [Banco de Dados do Azure para PostgreSQL - Servidor Flexível](#).
- Um sistema cliente PostgreSQL para executar os comandos despejo e restauração.
 - Pode ser um cliente Linux ou Windows com PostgreSQL instalado e tem os utilitários de linha de comando [pg_dump](#) e [pg_restore](#) instalados.
 - Como alternativa, você pode usar o [Azure Cloud Shell](#) ou clicando no Azure Cloud Shell na barra de menus na parte superior direita do [portal do Azure](#). Você precisará fazer logon em sua conta `az login` antes de executar os comandos despejo e restauração.
- O cliente PostgreSQL preferencialmente executado na mesma região que os servidores de origem e de destino.

Detalhes e considerações adicionais

- Você pode encontrar a cadeia de conexão para os bancos de dados de origem e de destino clicando em "Cadeias de Conexão" no portal.
- Você pode executar mais de um banco de dados em seu servidor. Você pode encontrar a lista de bancos de dados conectando-se ao seu servidor de origem e executando `\l`.
- Crie bancos de dados correspondentes no servidor de banco de dados de destino.
- Você pode ignorar a atualização `azure_maintenance` ou os bancos de dados de modelo.
- Confira as tabelas acima para determinar se o banco de dados é adequado para esse modo de migração.
- Se você quiser usar o Azure Cloud Shell, observe que a sessão atinge o tempo limite após 20 minutos. Se o tamanho do seu banco de dados for < 10 GB, você poderá concluir a atualização sem atingir o tempo limite da sessão. Caso contrário, talvez você precise manter a sessão aberta por outros meios, como pressionar a tecla uma vez a cada 10-15 minutos.

Banco de dados de exemplo usado neste guia

Neste guia, os servidores de origem, de destino e os nomes de banco de dados a seguir são usados para ilustrar com exemplos.

DESCRIÇÃO	VALOR
Servidor de origem (v9.5)	<code>pg-95.postgres.database.azure.com</code>
Banco de dados de origem	<code>bench5gb</code>

Descrição	Valor
Tamanho do banco de dados de origem	5 GB
Nome de usuário de origem	pg@pg-95
Servidor de destino (v11)	pg-11.postgres.database.azure.com
Banco de dados de destino	bench5gb
Nome de usuário de destino	pg@pg-11

NOTE

O servidor flexível dá suporte ao PostgreSQL versão 11 em diante. Além disso, o nome de usuário do servidor flexível não requer @.

Atualizar seus bancos de dados usando métodos de migração offline

Você pode optar por usar um dos métodos descritos nesta seção para suas atualizações. Você pode usar as dicas a seguir ao executar as tarefas.

- Se você estiver usando a mesma senha para o banco de dados de origem e destino, poderá definir a variável de ambiente `PGPASSWORD=yourPassword`. Assim você não precisará fornecer a senha toda vez que executar comandos como `psql`, `pg_dump` e `pg_restore`. Da mesma forma, você pode configurar variáveis adicionais como `PGUSER`, `PGSSLMODE` etc. Confira [Variáveis de ambiente do PostgreSQL](#).
- Se o seu servidor PostgreSQL exigir conexões TLS/SSL (ativas por padrão em servidores do Banco de Dados do Azure para PostgreSQL), defina uma variável de ambiente `PGSSLMODE=require` para que a ferramenta de `pg_restore` se conecte com TLS. Sem TLS, o erro pode indicar


```
FATAL: SSL connection is required. Please specify SSL options and retry.
```
- Na linha de comando do Windows, execute o comando `SET PGSSLMODE=require` antes de executar o comando `pg_restore`. No Linux ou o Bash, execute o comando `export PGSSLMODE=require` antes de executar o comando `pg_restore`.

Método 1: migrar usando o arquivo de despejo

Esse método envolve duas etapas. Primeiro, criar um despejo do servidor de origem. A segunda etapa é restaurar o arquivo de despejo para o servidor de destino. Para obter mais detalhes, confira a documentação [Migrar usando despejo e restauração](#). Esse é o método recomendado se você tiver grandes bancos de dados e o sistema cliente tiver armazenamento suficiente para armazenar o arquivo de despejo.

Método 2: migrar usando o streaming dos dados de despejo para o banco de dado de destino

Se você não tiver um cliente PostgreSQL ou quiser usar o Azure Cloud Shell, poderá usar esse método. O despejo do banco de dados é transmitido diretamente para o servidor de banco de dados de destino e não armazena o despejo no cliente. Portanto, isso pode ser usado com um cliente com armazenamento limitado e até mesmo pode ser executado a partir do Azure Cloud Shell.

- Verifique se o banco de dados existe no servidor de destino usando o comando `\l`. Se o banco de dados não existir, crie o banco de dados.

```
psql "host=myTargetServer port=5432 dbname=postgres user=myUser password=##### sslmode=mySSLmode"
```

```
postgres> \l
postgres> create database myTargetDB;
```

2. Execute o despejo e a restauração como uma linha de comando única usando um pipe.

```
pg_dump -Fc -v --mySourceServer --port=5432 --username=myUser --dbname=mySourceDB | pg_restore -v --no-owner --host=myTargetServer --port=5432 --username=myUser --dbname=myTargetDB
```

Por exemplo,

```
pg_dump -Fc -v --host=pg-95.postgres.database.azure.com --port=5432 --username=pg@pg-95 --dbname=bench5gb | pg_restore -v --no-owner --host=pg-11.postgres.database.azure.com --port=5432 --username=pg@pg-11 --dbname=bench5gb
```

3. Depois que o processo de atualização (migração) for concluído, você poderá testar seu aplicativo com o servidor de destino.

4. Repita esse processo para todos os bancos de dados no servidor.

Por exemplo, a tabela a seguir ilustra o tempo necessário para migrar usando o método de despejo de streaming. Os dados de exemplo são populados usando [pgbench](#). Como seu banco de dados pode ter um número diferente de objetos com tamanhos variados do que tabelas e índices gerados pelo pgbench, é altamente recomendável testar o despejo e a restauração do banco de dados para entender o tempo real necessário para atualizar seu banco de dados.

TAMANHO DO BANCO DE DADOS	TEMPO GASTO APROXIMADAMENTE
1 GB	1 – 2 minutos
5 GB	8 – 10 minutos
10 GB	15-20 minutos
50 GB	1 – 1,5 horas
100 GB	2,5 – 3 horas

Método 3: migrar usando o despejo e a restauração paralelos

Você pode considerar esse método se tiver poucas tabelas grandes no banco de dados e quiser paralelizar o processo de despejo e restauração para esse banco de dados. Você também precisa de armazenamento suficiente no seu sistema cliente para acomodar os despejos de backup. Esse processo paralelo de despejo e restauração reduz o tempo de consumo para concluir a migração inteira. Por exemplo, o banco de dados de 50 GB pgbench que levou de 1 a 1,5 h para migrar foi concluído usando o método 1 e 2, levou menos de 30 minutos usando esse método.

1. Para cada banco de dados no servidor de origem, crie um banco de dados correspondente no servidor de destino.

```
psql "host=myTargetServer port=5432 dbname=postgres user=myuser password=##### sslmode=mySSLmode"
postgres> create database myDB;
```

Por exemplo,

```
psql "host=pg-11.postgres.database.azure.com port=5432 dbname=postgres user=pg@pg-11 password=#####sslmode=require"

postgres> create database bench5gb;
postgres> \q
```

- Execute o comando pg_dump em um formato de diretório com número de trabalhos = 4 (número de tabelas no banco de dados). Com uma camada de computação maior e com mais tabelas, você pode aumentá-la para um número maior. Esse pg_dump criará um diretório para armazenar arquivos compactados para cada trabalho.

```
pg_dump -Fd -v --host=sourceServer --port=5432 --username=myUser --dbname=mySourceDB -j 4 -f myDumpDirectory
```

Por exemplo,

```
pg_dump -Fd -v --host=pg-95.postgres.database.azure.com --port=5432 --username=pg@pg-95 --dbname=bench5gb -j 4 -f dump.dir
```

- Em seguida, restaure o backup no servidor de destino.

```
$ pg_restore -v --no-owner --host=myTargetServer --port=5432 --username=myUser --dbname=myTargetDB -j 4 myDumpDir
```

Por exemplo,

```
$ pg_restore -v --no-owner --host=pg-11.postgres.database.azure.com --port=5432 --username=pg@pg-11 --dbname=bench5gb -j 4 dump.dir
```

TIP

O processo mencionado neste documento também pode ser usado para atualizar seu Banco de Dados do Azure para PostgreSQL – Servidor flexível, que está em versão prévia. A principal diferença é que a cadeia de conexão para o servidor flexível de destino não tem o `@dbName`. Por exemplo, se o nome de usuário for `pg`, o nome de usuário do servidor único na cadeia de conexão será `pg@pg-95`, enquanto que com o servidor flexível, você poderá simplesmente usar `pg`.

Próximas etapas

- Quando ficar satisfeito com a função de banco de dados de destino, você poderá remover o servidor de banco de dados antigo.
- Somente para o Banco de Dados do Azure para PostgreSQL - Servidor Único. Se quiser usar o mesmo ponto de extremidade de banco de dados que o servidor de origem depois de excluir o servidor de banco de dados de origem antigo, você poderá criar uma réplica de leitura com o nome do servidor de banco de dados antigo. Quando o estado de replicação estável for estabelecido, você poderá interromper a réplica, o que promoverá o servidor de réplica para ser um servidor independente. Consulte [Replicação](#) para obter mais detalhes.
- Lembre-se de testar e validar esses comandos em um ambiente de teste antes de usá-los em produção.

Banco de Dados do Azure para PostgreSQL – Problemas conhecidos e limitações

21/05/2021 • 2 minutes to read

Esta página fornece uma lista de problemas conhecidos do Banco de Dados do Azure para PostgreSQL que podem afetar seu aplicativo. Também lista opções de mitigação e recomendações para solucionar o problema.

Desempenho inteligente – Repositório de Consultas

Aplicável ao Banco de Dados do Azure para PostgreSQL – servidor único.

APLICÁVEL	CAUSA	REMEDIAÇÃO
PostgreSQL 9.6, 10 e 11	A ativação do parâmetro do servidor <code>pg_qs.replace_parameter_placeholders</code> pode levar a um desligamento do servidor em alguns cenários raros.	Pelo portal do Azure, na seção Parâmetros do Servidor, transforme o valor do parâmetro <code>pg_qs.replace_parameter_placeholders</code> em <code>OFF</code> e salve-o.

Parâmetros do Servidor

Aplicável ao Banco de Dados do Azure para PostgreSQL – servidor único e servidor flexível

APLICÁVEL	CAUSA	REMEDIAÇÃO
PostgreSQL 9.6, 10 e 11	A alteração do parâmetro do servidor <code>max_locks_per_transaction</code> para um valor mais alto do que o recomendado pode levar à não inicialização do servidor após uma reinicialização.	Mantenha-o com o valor padrão (32 ou 64) ou altere-o para um valor razoável de acordo com a documentação do PostgreSQL. No lado do serviço, isso está sendo trabalhado para limitar o valor alto com base no SKU.

Próximas etapas

- Confira [Melhores práticas para o Repositório de Consultas](#)

Servidor Único do Banco de Dados do Azure para PostgreSQL

21/05/2021 • 7 minutes to read

O [Banco de Dados do Azure para PostgreSQL](#) desenvolvido com o PostgreSQL community edition está disponível em três modos de implantação:

- Servidor único
- Servidor Flexível (versão prévia)
- Hiperescala (Citus)

Neste artigo, forneceremos uma visão geral e introdução aos conceitos principais do modelo de implantação de servidor único. Para saber mais sobre o modo de implantação de servidor flexível, confira a [visão geral do servidor flexível](#) e a Visão Geral de Hiperescala (Citus), respectivamente.

Visão geral

O Servidor Único é um serviço de banco de dados totalmente gerenciado com requisitos mínimos para personalizações do banco de dados. A plataforma de servidor único é projetada para administrar a maioria das funções de gerenciamento de banco de dados, como aplicação de patches, backups, alta disponibilidade, controle e segurança com mínima configuração do usuário. A arquitetura é otimizada para fornecer 99,99% de disponibilidade em uma única zona de disponibilidade. Ela dá suporte à versão da comunidade do PostgreSQL 9.5, 9.6, 10 e 11. O serviço está em disponibilidade geral em diversas [regiões do Azure](#).

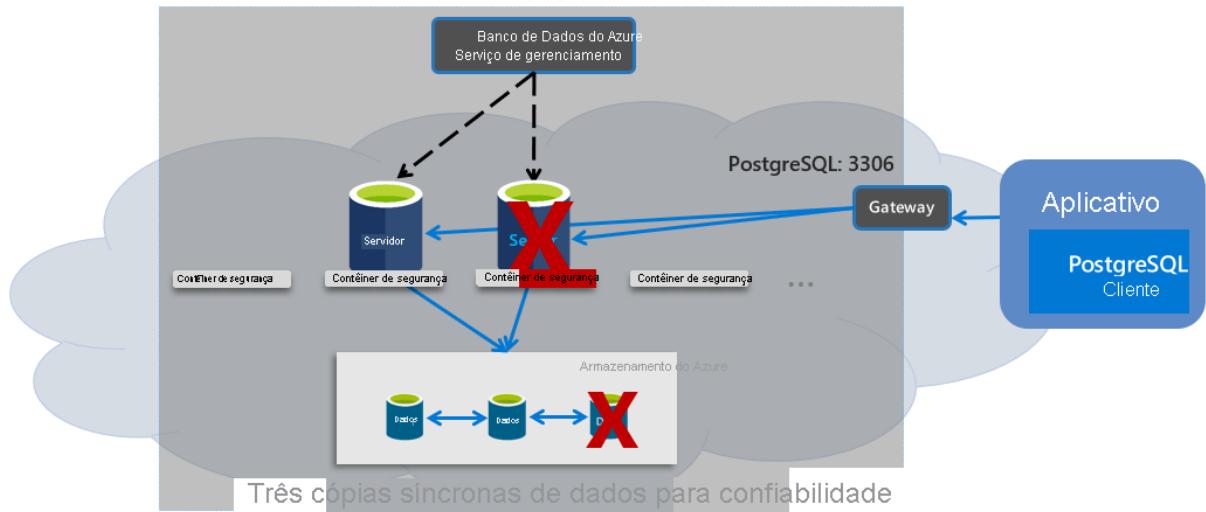
Servidores únicos são mais adequados para aplicativos nativos na nuvem projetados para lidar com aplicação de patch automatizada sem a necessidade de controle granular sobre o agendamento de aplicação de patch e definições de configuração de PostgreSQL personalizadas.

Alta disponibilidade

O modelo de implantação de servidor único é otimizado para alta disponibilidade interna e elasticidade a um custo reduzido. A arquitetura separa a computação do armazenamento. O mecanismo de banco de dados é executado em um contêiner de computação proprietário, enquanto os arquivos de dados são residem no armazenamento do Azure. O armazenamento mantém três cópias síncronas localmente redundantes dos arquivos de banco de dados, garantindo a durabilidade dos dados.

Durante eventos de failover planejados ou não planejados, se o servidor ficar inativo, o serviço manterá a alta disponibilidade dos servidores usando o seguinte procedimento automatizado:

1. Um novo contêiner de computação é provisionado.
2. O armazenamento com arquivos de dados é mapeado para o novo contêiner.
3. O mecanismo de banco de dados do PostgreSQL é colocado online no novo contêiner de computação.
4. O serviço de gateway garante um failover transparente, garantindo que nenhuma alteração no lado do aplicativo seja exigida.



O tempo de failover típico varia de 60 a 120 segundos. O design nativo de nuvem do serviço de servidor único permite que ele dê suporte a 99,99% da disponibilidade, eliminando o custo de espera ativa/passiva.

O SLA (contrato de nível de serviço) de disponibilidade de 99,99% do Azure, que é líder do setor e é desenvolvido por uma rede global de datacenters gerenciados pela Microsoft, ajuda a manter os seus aplicativos em execução de modo ininterrupto.

Aplicação de patch automatizada

O serviço executa a aplicação automatizada de patch do hardware, do sistema operacional e do mecanismo de banco de dados subjacentes. A aplicação de patch inclui atualizações de segurança e software. Para o mecanismo do PostgreSQL, as atualizações de versão secundária são automáticas e incluídas como parte do ciclo de aplicação de patch. Não há nenhuma ação do usuário nem definições de configuração necessárias para a aplicação de patch. A frequência de aplicação de patch é gerenciada pelo serviço com base na importância do conteúdo. No geral, o serviço segue a agenda de lançamento mensal como parte do lançamento e da integração contínua. Os usuários podem assinar a [notificação de manutenção planejada](#) para receber a notificação das próximas 72 horas de manutenção antes do evento.

Backups automáticos

O serviço de servidor único cria backups de servidor automaticamente e os armazena no LRS (armazenamento com redundância local) ou no armazenamento com redundância geográfica configurado pelo usuário. Os backups podem ser usados para restaurar o servidor em qualquer ponto no tempo dentro do período de retenção de backup. O período de retenção de backup padrão é de sete dias. A retenção pode ser configurada opcionalmente em até 35 dias. Todos os backups são criptografados usando a criptografia AES de 256 bits. Confira [Backups](#) para obter detalhes.

Ajustar o desempenho e a escala em segundos

O serviço de servidor único está disponível em três níveis de SKU: Básico, Uso Geral e Otimizado para Memória. O nível Básico é mais adequado para o desenvolvimento de baixo custo e cargas de trabalho com baixa simultaneidade. O Uso Geral e o Otimizado para Memória são mais adequados para cargas de trabalho de produção que exigem alta simultaneidade, escala e desempenho previsível. Você pode criar seu primeiro aplicativo em um banco de dados pequeno por alguns dólares por mês e então ajustar a escala para atender às necessidades da sua solução. A escala do armazenamento está online e dá suporte ao aumento automático do armazenamento. A escalabilidade dinâmica permite que o banco de dados responda de forma transparente a mudanças rápidas nos requisitos de recursos. Você paga somente pelos recursos que consome. Veja [Tipos de preço](#) para obter detalhes.

Segurança, conformidade e governança de nível empresarial

O serviço de servidor único usa o módulo de criptografia validado por FIPS 140-2 para criptografia de armazenamento de dados em repouso. Os dados, incluindo backups, e os arquivos temporários criados durante a execução de consultas são criptografados. O serviço usa a criptografia AES de 256 bits incluída na criptografia de armazenamento do Azure, e as chaves são gerenciadas pelo sistema (padrão) ou [gerenciadas pelo cliente](#). O serviço criptografa os dados em movimento com o protocolo SSL/TLS imposto por padrão. O serviço dá suporte às versões do TLS 1.2, 1.1 e 1.0 com uma capacidade de impor a [versão de TLS mínima](#).

O serviço permite o acesso privado aos servidores usando o link privado e fornece o recurso Proteção avançada contra ameaças. A proteção avançada contra ameaças detecta atividades anômalas que indicam tentativas incomuns e potencialmente prejudiciais de acesso ou exploração dos bancos de dados.

Além da autenticação nativa, o serviço de servidor único dá suporte à autenticação do Azure Active Directory. A autenticação do Azure AD é um mecanismo de conexão com os servidores PostgreSQL usando identidades definidas e gerenciadas no Azure AD. Com a autenticação do Azure AD, é possível gerenciar as identidades de usuários do banco de dados e outros serviços do Azure em uma localização central, o que simplifica e centraliza o controle de acesso.

O [log de auditoria](#) (em versão prévia) está disponível para rastrear toda a atividade no nível do banco de dados.

O serviço de servidor único está em conformidade com todas as certificações líderes do setor, como FedRAMP, HIPAA, PCI DSS. Visite a [Central de Confiabilidade do Azure](#) para obter informações sobre a segurança da plataforma do Azure.

Para obter mais informações sobre os recursos de segurança do Banco de Dados do Azure para PostgreSQL, confira a [visão geral de segurança](#).

Monitoramento e alertas

O serviço de servidor único é equipado com recursos internos de monitoramento e alerta de desempenho. Todas as métricas do Azure têm uma frequência de um minuto e cada uma delas fornece 30 dias de histórico. É possível configurar alertas nas métricas. O serviço permite configurar logs de consulta lentas e vem com um recurso [Repositório de consultas](#) diferenciado. O Repositório de Consultas simplifica a solução de problemas ajudando você a rapidamente localizar as consultas de execução mais longa e que consomem mais recursos. Usando essas ferramentas, você pode otimizar rapidamente suas cargas de trabalho e configurar seu servidor para ter o melhor desempenho. Confira [Monitoramento](#) para obter mais detalhes.

Migração

O serviço executa a versão da comunidade do PostgreSQL. Isso permite a compatibilidade total do aplicativo e exige um custo de refatoração mínimo para migrar o aplicativo existente desenvolvido no mecanismo PostgreSQL para um serviço de servidor único. A migração para o servidor único pode ser executada usando uma das seguintes opções:

- **Despejo e restauração:** nas migrações offline, em que os usuários podem ter algum tempo de inatividade, realizar o despejo e a restauração com ferramentas da comunidade, como pg_dump e pg_restore, pode fornecer uma forma mais rápida de migração. Confira [Migrar usando despejo e restauração](#) para obter detalhes.
- **Serviço de Migração de Banco de Dados do Azure** – Para migrações integradas e simplificadas para o servidor único com um tempo de inatividade mínimo, você pode aproveitar o Serviço de Migração de Banco de Dados do Azure. Confira [DMS por meio do portal](#) e [DMS por meio da CLI](#).

Contatos

Para perguntas ou sugestões sobre como trabalhar com o Banco de Dados do Azure para PostgreSQL, envie um email para a equipe do Banco de Dados do Azure para PostgreSQL ([@Ask Banco de Dados do Azure para PostgreSQL](#)). Esse endereço de email não é um alias de suporte técnico.

Além disso, considere os seguintes pontos de contato, conforme apropriado:

- Para entrar em contato com o Suporte do Azure, [crie um tíquete no Portal do Azure](#).
- Para corrigir um problema com sua conta, apresente uma [solicitação de suporte](#) no portal do Azure.
- Para fornecer comentários ou solicitar novos recursos, crie uma entrada por meio do [UserVoice](#).

Próximas etapas

Agora que você leu a introdução ao modo de implantação do servidor único do Banco de Dados do Azure para PostgreSQL, você está pronto para:

- Criar o seu primeiro servidor.

Início Rápido: Criar um servidor do Banco de Dados do Azure para PostgreSQL usando o portal do Azure

21/05/2021 • 5 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço gerenciado usado para executar, gerenciar e dimensionar bancos de dados altamente disponíveis do PostgreSQL na nuvem. Este guia de início rápido mostra como criar um Banco de Dados do Azure para PostgreSQL e conectar-se a ele.

Pré-requisitos

Será necessário ter uma assinatura ativa do Azure. Caso você não tenha uma assinatura do Azure, crie uma [conta gratuita do Azure](#) antes de começar.

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Acesse o [portal do Azure](#) para criar um banco de dados de Servidor Único do Banco de Dados do Azure para PostgreSQL. Pesquise e selecione *servidores do Banco de Dados do Azure para PostgreSQL*.

The screenshot shows the Azure portal search interface. The search bar at the top contains the text 'postgres'. Below the search bar, there are two main sections: 'Services' and 'Marketplace'.

Services section:

- Azure Database for PostgreSQL servers (selected)
- Azure Database for PostgreSQL server groups
- Azure Database for PostgreSQL servers v2
- Azure Database for PostgreSQL server groups - Azure Arc

Marketplace section:

- postgres
- Postgres Pro Enterprise Database 10 (VM)
- Postgres Pro Standard Database 9.6 (VM)
- Postgres Pro Standard Database 12 (VM)

Resources section:

No results were found.

Documentation section:

- [Azure Database for PostgreSQL documentation](#)
- [Tutorial: Design an Azure Database for PostgreSQL](#)
- [What is Azure Database for PostgreSQL | Microsoft Docs](#)
- [Logs - Azure Database for PostgreSQL - Single \\$](#)

Resource Groups section:

- freebergpostgrestomcat

Searchina 7 of 44 subscriptions. [Change](#)

1. Selecione **Adicionar**.
2. Na página Criar um Banco de Dados do Azure para PostgreSQL, selecione **Servidor único**.

Select Azure Database for PostgreSQL deployment option

Microsoft

How do you plan to use the service?



Single server

Best for broad range of traditional transactional workloads.

Enterprise ready, fully managed community PostgreSQL server with up to 64 vCores, optional geospatial support, full-text search and more.

[Create](#)[Learn more](#)

Flexible server (Preview)

Best for workloads that require advanced customization and cost optimization.

Maximum control with a simplified developer experience. Supports custom maintenance windows, zone redundant high availability, and simple cost optimization. Flexible server is currently in preview.

[Create](#)[Learn more](#)

Hyperscale (Citus) server group

Best for ultra-high performance and data needs beyond 100GB.

Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response. Supports both transactional/operational workloads and hybrid transactional analytics workloads.

[Create](#)[Learn more](#)

Azure Arc enabled PostgreSQL Hyperscale (Preview)

Best for ultra-high performance and data needs beyond 100GB on your infrastructure.

Deployed on the infrastructure of your choice(on-premises/edge/multi-cloud), it is ideal for multi-tenant applications, transactional/operational workloads and real-time analytical workloads that need sub-second response.

[Learn more](#)

3. Preencha o formulário **Informações Básicas** com as informações a seguir.

Single server

Microsoft

Basics Additional settings Tags Review + create

Create an Azure Database for PostgreSQL server. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Resource group * ⓘ

Select a resource group

[Create new](#)

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name * ⓘ

Data source * ⓘ

None Backup

Location * ⓘ

(US) East US

Version * ⓘ

10

Compute + storage ⓘ

General Purpose

4 vCores, 100 GB storage

[Configure server](#)

Administrator account

Admin username * ⓘ

Password * ⓘ

Confirm password *

[Review + create](#)

[Next : Additional settings >](#)

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Subscription	nome da sua assinatura	selecione a Assinatura do Azure desejada.
Resource group	myresourcegroup	um grupo de recursos novo ou existente da sua assinatura.

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Nome do servidor	<i>mydemoserver</i>	Um nome exclusivo que identifica o Banco de Dados do Azure para o servidor PostgreSQL. O nome de domínio <i>postgres.database.azure.com</i> é acrescentado ao nome do servidor fornecido por você. O servidor pode conter apenas letras minúsculas, números e o caractere de hífen (-). Ele precisa conter de 3 a 63 caracteres.
Fonte de dados	Nenhum	Selecione Nenhum para criar um novo servidor do zero. Selecione Backup somente se você estivesse restaurando de um backup geográfico de um servidor existente.
Nome de usuário do administrador	<i>myadmin</i>	Insira o nome do usuário administrador do servidor. Ele não pode começar com pg_ e estes valores não são permitidos: <i>azure_superuser</i> , <i>azure_pg_admin</i> , <i>admin</i> , <i>administrator</i> , <i>root</i> , <i>guest</i> ou <i>public</i> .
Senha	sua senha	Uma nova senha para o usuário administrador do servidor. Ela precisa conter de 8 a 128 caracteres de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números (0 a 9) e caracteres não alfanuméricos (por exemplo, !, \$, # e %).
Location	o local desejado	Selecione uma localização na lista suspensa.
Versão	A última versão principal	A última versão principal do PostgreSQL, a menos que você tenha requisitos específicos.
Computação + armazenamento	<i>usar os padrões</i>	O tipo de preço padrão é Uso Geral com 4 vCores e armazenamento de 100 GB . A retenção de backup é definida como 7 dias com opção de backup Geograficamente Redundante . Saiba mais sobre os preços e atualize os padrões, se necessário.

NOTE

Considere usar o tipo de preço Básico se computação leve e E/S forem adequadas para sua carga de trabalho. Observe que os servidores criados no tipo de preço Básico não podem ser escalados posteriormente para Uso Geral ou Otimizado para Memória.

4. Selecione **Revisar + criar** para revisar suas seleções. Selecione **Criar** para provisionar o servidor. Essa operação poderá levar alguns minutos.

NOTE

Um banco de dados vazio, **postgres**, é criado. Você também encontrará um banco de dados **azure_maintenance** que é usado para separar os processos de serviço gerenciado das ações do usuário. Não é possível acessar o banco de dados **azure_maintenance**.

The screenshot shows the Azure portal interface for a PostgreSQL deployment. The top navigation bar includes 'Home >', the deployment name 'Microsoft.PostgreSQLServer.createPostgreSQLServer_5615c1da8aa144 | Overview', and standard navigation buttons like 'Delete', 'Cancel', 'Redeploy', and 'Refresh'. On the left, a sidebar lists 'Overview' (selected), 'Inputs', 'Outputs', and 'Template'. The main content area displays a green checkmark icon and the message 'Your deployment is complete'. Below this, deployment details are shown: 'Deployment name: Microsoft.PostgreSQLServer.createPostgreSQLSe...', 'Subscription: mysubscriptionname', 'Resource group: mydemoserver', 'Start time: 10/20/2020, 5:27:11 PM', and 'Correlation ID: b492c68b-955d-4276-8739-41f450919169'. A 'Deployment details' link with a download icon is available. A table summarizes the resources: Resource (mydemoserver), Type (Microsoft.DBforPostgreSQL/servers), Status (Created), and Operation details (link). Below the table, a 'Next steps' section contains a 'Go to resource' button.

Está com problemas? Fale conosco.

Configurar uma regra de firewall

Por padrão, o servidor que você cria não é publicamente acessível. Você precisa conceder permissões ao seu endereço IP. Acesse o recurso de servidor no portal do Azure e selecione **Segurança da conexão** no menu do lado esquerdo do recurso de servidor. Caso não tenha certeza de como localizar seu recurso, confira [Abrir recursos](#).

The screenshot shows the 'Connection security' settings for a PostgreSQL server. The left sidebar includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', and 'Settings' (selected). Under 'Settings', there are sections for 'Connection security' (selected), 'Connection strings', 'Server parameters', 'Active Directory admin', 'Pricing tier', 'Properties', 'Locks', and 'Export template'. The main content area shows the 'Firewall rules' section. It includes a note about connections from specified IPs providing access to all databases. A toggle switch for 'Allow access to Azure services' is set to 'Yes'. Buttons for '+ Add current client IP address (24.16.139.249)' and '+ Add 0.0.0.0 - 255.255.255.255' are present. A table for 'Firewall rule name' has columns for 'Firewall rule name', 'Start IP', and 'End IP'. The 'SSL settings' section shows 'Enforcing SSL connections on your server may require additional configuration to your applications connecting to the server.' A toggle switch for 'Enforce SSL connection' is set to 'ENABLED'.

Selecione **Adicionar o endereço IP do cliente atual** e escolha **Salvar**. Adicione mais endereços IP ou forneça um intervalo de IP para se conectar ao seu servidor nesses endereços IP. Para obter mais informações, confira [Regras de firewall do Banco de Dados do Azure para PostgreSQL](#).

NOTE

Para evitar problemas de conectividade, verifique se a sua rede permite o tráfego de saída na porta 5432. O Banco de Dados do Azure para PostgreSQL usa essa porta.

Está com problemas? Fale conosco.

Conectar-se ao servidor com psql

Use o [psql](#) ou o [pgAdmin](#), que são clientes populares do PostgreSQL. Neste guia de início rápido, vamos nos conectar usando o psql no [Azure Cloud Shell](#) dentro do portal do Azure.

1. Anote o nome do servidor, o nome de logon do administrador do servidor, a senha e a ID da assinatura do servidor recém-criado na seção **Visão geral** do servidor.

Nome do servidor : mydemoserver.postgres.database.azure.com
Nome de usuário do administrador : demoadmin@mydemoserver
Versão do PostgreSQL : 10
Configuração do desempenho: Uso geral, 4 vCore(s), 100 GB
Status de imposição do SSL : HABILITADO

2. Abra o Azure Cloud Shell no portal selecionando o ícone no lado superior esquerdo.

NOTE

Se você estiver abrindo o Cloud Shell pela primeira vez, verá um aviso que solicitará a criação de um grupo de recursos e uma conta de armazenamento. Essa é uma etapa única e será anexada automaticamente para todas as sessões.

```
Bash
Requesting a Cloud Shell. Succeeded.
Connecting terminal...
Welcome to Azure Cloud Shell
Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell
sunitha@Azure:~$
```

3. Execute o comando a seguir no terminal do Azure Cloud Shell. Substitua os valores pelos nomes reais do servidor e de logon do usuário administrador. Use o banco de dados vazio **postgres** com o usuário

administrador neste formato: <admin-username>@<servername> .

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --dbname=postgres
```

Veja como é a experiência no terminal do Cloud Shell:

```
Requesting a Cloud Shell.Succeeded.  
Connecting terminal...  
  
Welcome to Azure Cloud Shell  
  
Type "az" to use Azure CLI  
Type "help" to learn about Cloud Shell  
  
user@Azure:~$psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --dbname=postgres  
Password for user myadmin@mydemoserver.postgres.database.azure.com:  
psql (12.2 (Ubuntu 12.2-2.pgdg16.04+1), server 11.6)  
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)  
Type "help" for help.  
  
postgres=>
```

4. No mesmo terminal do Azure Cloud Shell, crie um banco de dados chamado **convidado**.

```
postgres=> CREATE DATABASE guest;
```

5. Alterne as conexões para o banco de dados **convidado** recém-criado.

```
\c guest
```

6. Digite **\q** e selecione a tecla ENTER para fechar o psql.

[Está com problemas? Fale conosco.](#)

Limpar os recursos

Você criou com êxito um servidor do Banco de Dados do Azure para PostgreSQL em um grupo de recursos. Se você não espera precisar desses recursos no futuro, exclua-os eliminando o grupo de recursos ou o servidor PostgreSQL.

Para excluir o grupo de recursos:

1. No portal do Azure, procure por **Grupos de recursos** e selecione essa opção.
2. Na lista grupo de recursos, escolha o nome do seu grupo de recursos.
3. Na página **Visão geral** do grupo de recursos, selecione **Excluir grupo de recursos**.
4. Na caixa de diálogo de confirmação, insira o nome do seu grupo de recursos e selecione **Excluir**.

Para excluir o servidor, selecione o botão **Excluir** na página **Visão geral** do servidor:

The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'mydemouser'. On the left, there's a sidebar with options like 'Create a resource', 'All services', 'FAVORITES' (Dashboard, Resource groups), 'All resources', 'Recent', and 'Subscriptions'. The main area is titled 'mydemouser' and 'Azure Database for PostgreSQL server'. It has tabs for 'Overview' (which is selected and highlighted in blue), 'Activity log', 'Tags', 'SETTINGS' (with 'Connection security', 'Connection strings', and 'Generate instrumentation' options), and 'Monitoring'. At the top right of the main content area, there are buttons for 'Reset password', 'Restore', and 'Delete', with the 'Delete' button being the one highlighted by a red box. Below these buttons, there's a section titled 'Essential' with details about the resource group ('myresourcegroup'), status ('Available'), location ('Southeast Asia'), subscription name ('Free Trial'), and subscription ID ('0000000-0000-0000-0000-000000000000'). To the right of these details, there's a summary of server configuration: 'Server name' (mydemouser.postgres.database.azure.com), 'Server admin login name' (myadmin@mydemouser), 'PostgreSQL version' (9.6), 'Performance configuration' (General Purpose, 2 vCore(s), 5 GB), and 'SSL enforce status' (ENABLED).

Está com problemas? Fale conosco.

Próximas etapas

[Migrar seu banco de dados usando a exportação e a importação](#)

[Criar um banco de dados](#)

Não foi possível encontrar o que estava procurando? Fale conosco.

Início Rápido: Criar um servidor do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure

21/05/2021 • 5 minutes to read

Este guia de início rápido mostra como usar os comandos da [CLI do Azure](#) no [Azure Cloud Shell](#) para criar um servidor único do Banco de Dados do Azure para PostgreSQL em cinco minutos. Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Pré-requisitos

- Use o ambiente Bash no [Azure Cloud Shell](#).
[Launch Cloud Shell](#)
- Se preferir, [instale](#) a CLI do Azure para executar comandos de referência da CLI.
 - Se estiver usando uma instalação local, entre com a CLI do Azure usando o comando `az login`. Para concluir o processo de autenticação, siga as etapas exibidas no terminal. Para mais opções de entrada, confira [Entrar com a CLI do Azure](#).
 - Quando solicitado, instale as extensões da CLI do Azure no primeiro uso. Para obter mais informações sobre extensões, confira [Usar extensões com a CLI do Azure](#).
 - Execute `az version` para localizar a versão e as bibliotecas dependentes que estão instaladas. Para fazer a atualização para a versão mais recente, execute `az upgrade`.
- Este artigo exige a versão 2.0 ou posterior da CLI do Azure. Se você está usando o Azure Cloud Shell, a versão mais recente já está instalada.

TIP

Considere a possibilidade de usar o comando mais simples da CLI do Azure `az postgres up`, que atualmente está em versão prévia. Experimente o [início rápido](#).

- Selecione a ID da assinatura específica na sua conta usando o comando `az account set`.
 - Anote o valor de `id` da saída `az login` para usá-lo como o valor para o argumento `subscription` no comando.

```
az account set --subscription <subscription id>
```

- Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Para obter todas as suas assinaturas, use `az account list`.

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Crie um [grupo de recursos do Azure](#) usando o comando `az group create` e crie o servidor PostgreSQL dentro desse grupo de recursos. Você deve fornecer um nome exclusivo. O exemplo a seguir cria um grupo de recursos

chamado `myresourcegroup` na localização `westus`.

```
az group create --name myresourcegroup --location westus
```

Crie um **Servidor do Banco de Dados do Azure para PostgreSQL** usando o comando `az postgres server create`. Um servidor pode conter vários bancos de dados.

```
az postgres server create --resource-group myresourcegroup --name mydemoserver --location westus --admin-user myadmin --admin-password <server_admin_password> --sku-name GP_Gen5_2
```

Estes são os detalhes dos argumentos anteriores:

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
name	mydemoserver	Nome exclusivo que identifica o servidor do Banco de Dados do Azure para PostgreSQL. O nome do servidor pode conter apenas letras minúsculas, números e o caractere de hífen (-). Ele precisa conter de 3 a 63 caracteres. Para obter mais informações, confira Regras de nomenclatura do Banco de Dados do Azure para PostgreSQL .
resource-group	myresourcegroup	Nome do grupo de recursos do Azure.
local	westus	A localização do Azure para o servidor.
admin-user	myadmin	Nome de usuário para o logon de administrador. Ele não pode ser <code>azure_superuser</code> , <code>admin</code> , <code>administrator</code> , <code>root</code> , <code>guest</code> nem <code>public</code> .
admin-password	<i>senha de segurança</i>	Senha do usuário administrador. Ela precisa conter de 8 a 128 caracteres de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números e caracteres não alfanuméricos.
sku-name	GP_Gen5_2	Nome do tipo de preço e a configuração de computação. Segue a convenção {tipo de preço} {geração da computação} {vCores} em formato abreviado. Para obter mais informações, confira Preços do Banco de Dados do Azure para PostgreSQL .

IMPORTANT

- A versão padrão do PostgreSQL no seu servidor é 9.6. Para ver todas as versões compatíveis, confira [Versões principais compatíveis do PostgreSQL](#).
- Para ver todos os argumentos do comando `az postgres server create`, confira [este documento de referência](#).
- O SSL é habilitado por padrão no seu servidor. Para obter mais informações sobre o SSL, confira [Configurar a conectividade SSL](#).

Configurar uma regra de firewall no nível de servidor

Por padrão, o servidor criado não é publicamente acessível e é protegido com regras de firewall. Você pode configurar as regras de firewall no servidor usando o comando `az postgres server firewall-rule create` para dar acesso ao seu ambiente local para se conectar ao servidor.

O exemplo a seguir cria uma regra de firewall chamada `AllowMyIP`, que permite conexões de um endereço IP específico, 192.168.0.1. Substitua o endereço IP ou o intervalo dos endereços IP que correspondem ao local em que você se conectará. Se você não souber seu endereço IP, acesse WhatIsMyIPAddress.com para obtê-lo.

```
az postgres server firewall-rule create --resource-group myresourcegroup --server mydemoserver --name AllowMyIP --start-ip-address 192.168.0.1 --end-ip-address 192.168.0.1
```

NOTE

Para evitar problemas de conectividade, verifique se o firewall da rede permite a porta 5432. Os servidores do Banco de Dados do Azure para PostgreSQL usam essa porta.

Obter informações de conexão

Para se conectar ao servidor, forneça credenciais de acesso e informações do host.

```
az postgres server show --resource-group myresourcegroup --name mydemoserver
```

O resultado está no formato JSON. Anote os valores de `administratorLogin` e `fullyQualifiedDomainName`.

```
{
  "administratorLogin": "myadmin",
  "earliestRestoreDate": null,
  "fullyQualifiedDomainName": "mydemoserver.postgres.database.azure.com",
  "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myresourcegroup/providers/Microsoft.DBforPostgreSQL/servers/mydemoserver",
  "location": "westus",
  "name": "mydemoserver",
  "resourceGroup": "myresourcegroup",
  "sku": {
    "capacity": 2,
    "family": "Gen5",
    "name": "GP_Gen5_2",
    "size": null,
    "tier": "GeneralPurpose"
  },
  "sslEnforcement": "Enabled",
  "storageProfile": {
    "backupRetentionDays": 7,
    "geoRedundantBackup": "Disabled",
    "storageMb": 5120
  },
  "tags": null,
  "type": "Microsoft.DBforPostgreSQL/servers",
  "userVisibleState": "Ready",
  "version": "9.6"
}
```

Conectar-se ao servidor do Banco de Dados do Azure para PostgreSQL usando o psql

O cliente [psql](#) é uma opção popular para se conectar a servidores PostgreSQL. Conecte-se ao seu servidor usando o psql com o [Azure Cloud Shell](#). Como alternativa, use também o psql no ambiente local se ele estiver disponível. Um banco de dados vazio, o **postgres**, é criado automaticamente com um novo servidor PostgreSQL. Use esse banco de dados para se conectar ao psql, conforme mostrado no código a seguir.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --dbname=postgres
```

TIP

Se você preferir usar um caminho de URL para se conectar ao Postgres, codifique a URL de entrada @ do nome de usuário com `%40`. Por exemplo, a cadeia de conexão do psql será:

```
psql postgresql://myadmin%40mydemoserver@mydemoserver.postgres.database.azure.com:5432/postgres
```

Limpar os recursos

Caso você não precise desses recursos para outro guia de início rápido ou tutorial, exclua-os executando o comando a seguir.

```
az group delete --name myresourcegroup
```

Se você apenas deseja excluir o servidor recém-criado, execute o comando [az postgres server delete](#).

```
az postgres server delete --resource-group myresourcegroup --name mydemoserver
```

Próximas etapas

[Migrar seu banco de dados usando a exportação e a importação](#)

Início Rápido: Use um comando da CLI do Azure, az postgres up (versão prévia), para criar um Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 4 minutes to read

IMPORTANT

O comando da CLI do Azure `az postgres up` está na versão prévia.

O Banco de Dados do Azure para PostgreSQL é um serviço gerenciado que permite executar, gerenciar e dimensionar os bancos de dados altamente disponíveis do PostgreSQL na nuvem. A CLI do Azure é usada para criar e gerenciar recursos do Azure da linha de comando ou em scripts. Este guia de início rápido mostra como usar o comando `az postgres up` para criar um servidor Banco de Dados do Azure para PostgreSQL usando a CLI do Azure. Além de criar o servidor, o `az postgres up` comando cria um banco de dados de exemplo, um usuário raiz no banco de dados, abre o firewall para serviços do Azure e cria regras de firewall para o computador cliente de padrão. Esses padrões ajudam a acelerar o processo de desenvolvimento.

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Este artigo exige que você esteja executando a CLI do Azure versão 2.0 ou posterior localmente. Para ver a versão instalada, execute o comando `az --version`. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Você precisará entrar na sua conta usando o comando `az login`. Observe a propriedade ID da saída do comando para obter o nome da assinatura correspondente.

```
az login
```

Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Selecione a ID da assinatura específica em sua conta usando o comando `az account set`. Substitua a propriedade **ID da assinatura** da saída `az login` por sua assinatura no espaço reservado da ID de assinatura.

```
az account set --subscription <subscription id>
```

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Para usar os comandos, instale a extensão `db-up`. Para executar esta amostra, verifique se você instalou a última versão da CLI do Azure. Consulte [Instalar a CLI do Azure](#).

```
az extension add --name db-up
```

Criar um servidor Banco de Dados do Azure para PostgreSQL usando o comando a seguir:

```
az postgres up
```

O servidor é criado com os seguintes valores padrão (a menos que você os substitua manualmente):

CONFIGURAÇÃO	VALOR PADRÃO	DESCRIÇÃO
server-name	Gerada pelo sistema	Um nome exclusivo que identifica o Banco de Dados do Azure para o servidor PostgreSQL.
resource-group	Gerada pelo sistema	Um novo grupo de recursos do Azure.
sku-name	GP_Gen5_2	O nome da SKU. Segue a convenção {tipo de preço}_{geração de computação}_{vCores} em formato abreviado. O padrão é servidor Gen5 de Uso Geral com 2 vCores. Consulte nossa página de preços para obter mais informações sobre os tipos.
backup-retention	7	Por quanto tempo o backup é mantido. A unidade é dias.
geo-redundant-backup	Desabilitado	Indica se os backups com redundância geográfica devem ser habilitados para este servidor ou não.
local	westus2	O local do Azure para o servidor.
ssl-enforcement	Desabilitado	Se o TLS/SSL deve ser habilitado ou não para este servidor.
storage-size	5120	A capacidade de armazenamento do servidor (a unidade é megabytes).
version	10	A versão principal do PostgreSQL.
admin-user	Gerada pelo sistema	O nome de usuário para o administrador.
admin-password	Gerada pelo sistema	A senha do usuário administrador.

NOTE

Para obter mais informações sobre o comando `az postgres up` e seus parâmetros adicionais, consulte a [documentação da CLI do Azure](#).

Depois que o servidor é criado, ele vem com as seguintes configurações:

- É criada uma regra de firewall denominada "devbox". A CLI do Azure tenta detectar o endereço IP do computador no qual o comando `az postgres up` é executado e permite esse endereço IP.
- "Permitir acesso aos Serviços do Azure" está definido como ATIVADO. Isso configura o firewall do servidor para aceitar conexões de todos os recursos do Azure, incluindo os recursos que não estão na sua assinatura.
- É criado um banco de dados vazio chamado "sampledb"

- É criado um novo usuário denominado "raiz" com privilégios para "sampledb"

NOTE

O Banco de Dados do Azure para PostgreSQL se comunica pela porta 5432. Ao se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Solicite que seu departamento de TI abra a porta 5432 para se conectar ao seu servidor.

Obter informações de conexão

Após o comando `az postgres up` ser concluído, uma lista de cadeias de conexão para linguagens de programação populares é retornada. Essas cadeias de conexão são pré-configuradas com os atributos específicos de seu banco de dados recém-criados no servidor Banco de Dados do Azure para PostgreSQL.

Você pode usar o comando `az postgres show-connection-string` para listar essas cadeias de conexão novamente.

Limpar os recursos

Limpe todos os recursos que você criou no início rápido usando o comando a seguir. Esse comando exclui o servidor Banco de Dados do Azure para PostgreSQL e o grupo de recursos.

```
az postgres down --delete-group
```

Se você quiser simplesmente excluir o servidor recém-criado, poderá executar o comando `az postgres down`.

```
az postgres down
```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

Início Rápido: Criar um Banco de Dados do Azure para PostgreSQL – Servidor único usando o PowerShell

21/05/2021 • 10 minutes to read

Este início rápido descreve como usar o PowerShell para criar um servidor de Banco de Dados do Azure para PostgreSQL no grupo de recursos do Azure. Você pode usar o PowerShell para criar e gerenciar recursos do Azure interativamente ou em scripts.

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Se você optar por usar o PowerShell localmente, este artigo exigirá que você instale o módulo Az PowerShell e conecte-se à sua conta do Azure usando o cmdlet [Connect-AzAccount](#). Para obter mais informações sobre como instalar o módulo Az PowerShell, confira [Instalar o Azure PowerShell](#).

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Se esta é a primeira vez que você usa o serviço de Banco de Dados do Azure para PostgreSQL, registre o provedor de recursos **Microsoft.DBforPostgreSQL**.

```
Register-AzResourceProvider -ProviderNamespace Microsoft.DBforPostgreSQL
```

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	

OPÇÃO	EXEMPLO/LINK
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

Se tiver várias assinaturas do Azure, escolha a que for adequada para cobrança do recurso. Selecione uma ID de assinatura específica usando o cmdlet [Set-AzContext](#).

```
Set-AzContext -SubscriptionId 00000000-0000-0000-0000-000000000000
```

Criar um grupo de recursos

Crie um [grupo de recursos do Azure](#) usando o cmdlet [New-AzResourceGroup](#). Um grupo de recursos é um contêiner lógico no qual os recursos do Azure são implantados e gerenciados como um grupo.

O seguinte exemplo cria um grupo de recursos chamado **myresourcegroup** na região **Oeste dos EUA**.

```
New-AzResourceGroup -Name myresourcegroup -Location westus
```

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Crie um Banco de Dados do Azure para PostgreSQL com o cmdlet [New-AzPostgreSqlServer](#). Um servidor pode gerenciar vários bancos de dados. Normalmente, um banco de dados separado é usado para cada projeto ou para cada usuário.

A tabela a seguir contém uma lista de parâmetros usados com frequência e valores de exemplo para o cmdlet [New-AzPostgreSqlServer](#).

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
Nome	mydemoserver	Escolha um nome globalmente exclusivo no Azure que identifique o servidor do Banco de Dados do Azure para PostgreSQL. O nome do servidor pode conter apenas letras, números e o caractere de hífen (-). Todos os caracteres de letras maiúsculas especificados são convertidos automaticamente em de letras minúsculas durante o processo de criação. Ele deve conter de 3 a 63 caracteres.

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
ResourceGroupName	myresourcegroup	Forneça o nome do grupo de recursos do Azure.
Sku	GP_Gen5_2	O nome da SKU. Segue a convenção pricing-tier_compute-generation_vCores em resumo. Para obter mais informações sobre o parâmetro da SKU, confira as informações após esta tabela.
BackupRetentionDay	7	Quanto tempo um backup deve ser retido. A unidade é dias. O intervalo é de 7 a 35.
GeoRedundantBackup	habilitado	Indica se os backups com redundância geográfica devem ser habilitados para este servidor ou não. Esse valor não pode ser habilitado para servidores no tipo de preço básico e não pode ser alterado após a criação do servidor. Valores permitidos: Habilitado, Desabilitado.
Location	westus	A região do Azure para o servidor.
SslEnforcement	habilitado	Se o SSL deve ser habilitado para este servidor. Valores permitidos: Habilitado, Desabilitado.
StorageInMb	51200	A capacidade de armazenamento do servidor (a unidade é megabytes). O StorageInMb válido é um mínimo de 5120 MB e aumenta em incrementos de 1024 MB. Para obter mais informações sobre limites de tamanho de armazenamento, confira Tipos de preço do Banco de Dados do Azure para PostgreSQL .
Versão	9.6.	A versão principal do PostgreSQL.
AdministratorUserName	myadmin	O nome de usuário para o logon de administrador. Não pode ser azure_superuser, admin, administrator, root, guest nem public .
AdministratorLoginPassword	<code><securestring></code>	A senha do usuário administrador na forma de uma cadeia de caracteres segura. Ele deve conter entre 8 e 128 caracteres. A senha precisa conter caracteres de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números e caracteres não alfanuméricos.

O valor do parâmetro **SKU** segue a convenção **pricing-tier_compute-generation_vCores**, conforme

mostrado nestes exemplos.

- `-Sku B_Gen5_1` é mapeado para Básico, Gen 5 e 1 vCore. Essa opção é o menor SKU disponível.
- `-Sku GP_Gen5_32` mapeia para Uso Geral, Gen 5 e 32 vCores.
- `-Sku M0_Gen5_2` mapeia para Otimizado para Memória, Gen 5 e 2 vCores.

Para obter informações sobre valores de SKU válidos por região e para camadas, confira [Tipos de preço do Banco de Dados do Azure para PostgreSQL](#).

O exemplo a seguir cria um servidor PostgreSQL na região Oeste dos EUA denominada `mydemoserver` no grupo de recursos `myresourcegroup` com um logon de administrador do servidor de `myadmin`. É um servidor Gen 5 no tipo de preço de uso geral com dois vCores e backups com redundância geográfica habilitados. Documente a senha usada na primeira linha do exemplo, pois essa é a senha da conta do administrador do servidor PostgreSQL.

TIP

Um nome de servidor mapeia para um nome DNS e deve ser globalmente exclusivo no Azure.

```
$Password = Read-Host -Prompt 'Please enter your password' -AsSecureString  
New-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup -Sku GP_Gen5_2 -  
GeoRedundantBackup Enabled -Location westus -AdministratorUsername myadmin -AdministratorLoginPassword  
$Password
```

Considere usar o tipo de preço Básico se computação leve e E/S forem adequadas para sua carga de trabalho.

IMPORTANT

Os servidores criados no tipo de preço básico não podem depois ser dimensionados para uso geral ou com otimização de memória e não podem ser replicados geograficamente.

Configurar uma regra de firewall

Crie uma regra de firewall no nível de servidor do Banco de Dados do Azure para PostgreSQL usando o cmdlet `New-AzPostgreSqlServerRule`. Uma regra de firewall no nível de servidor permite que um aplicativo externo, como a ferramenta de linha de comando `psql` ou o PostgreSQL Workbench, conecte-se ao servidor por meio do firewall do serviço Banco de Dados do Azure para PostgreSQL.

O exemplo a seguir cria uma regra de firewall chamada `AllowMyIP`, que permite conexões de um endereço IP específico, 192.168.0.1. Substitua um endereço IP ou um intervalo de endereços IP que correspondam à localização da qual você está se conectando.

```
New-AzPostgreSqlServerRule -Name AllowMyIP -ResourceGroupName myresourcegroup -ServerName mydemoserver -  
StartIPAddress 192.168.0.1 -EndIPAddress 192.168.0.1
```

NOTE

As conexões ao Banco de Dados do Azure para PostgreSQL se comunicam pela porta 5432. Se estiver tentando se conectar em uma rede corporativa, talvez o tráfego de saída pela porta 5432 não seja permitido. Nesse cenário, você só poderá se conectar ao servidor se o departamento de TI abrir a porta 5432.

Obter informações de conexão

Para se conectar ao servidor, é preciso fornecer credenciais de acesso e informações do host. Use o exemplo a seguir para determinar as informações de conexão. Anote os valores de **FullyQualifiedDomainName** e o **AdministratorLogin**.

```
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |  
Select-Object -Property FullyQualifiedDomainName, AdministratorLogin
```

FullyQualifiedDomainName	AdministratorLogin
-----	-----
mydemoserver.postgres.database.azure.com	myadmin

Conectar-se ao banco de dados PostgreSQL usando psql

Se o computador cliente tiver o PostgreSQL instalado, você poderá usar uma instância local do `psql` para se conectar a um servidor PostgreSQL do Azure. Você também pode acessar uma versão pré-instalada da ferramenta de linha de comando `psql` no Azure Cloud Shell selecionando o botão **Experimentar** em um exemplo de código neste artigo. Outras maneiras de acessar o Azure Cloud Shell são selecionar o botão `>_` na barra de ferramentas superior direita no portal do Azure ou acessar shell.azure.com.

1. Conecte-se ao servidor PostgreSQL do Azure usando o utilitário de linha de comando `psql`.

```
psql --host=<servername> --port=<port> --username=<user@servername> --dbname=<dbname>
```

Por exemplo, o comando a seguir se conecta ao banco de dados padrão chamado `postgres` no seu servidor PostgreSQL `mydemoserver.postgres.database.azure.com` usando as credenciais de acesso. Insira o `<server_admin_password>` que você escolheu quando uma senha foi solicitada a você.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --  
dbname=postgres
```

TIP

Se você preferir usar um caminho de URL para se conectar ao Postgres, codifique a URL de entrada @ do nome de usuário com `%40`. Por exemplo, a cadeia de conexão para `psql` seria

```
psql postgresql://myadmin%40mydemoserver@mydemoserver.postgres.database.azure.com:5432/postgres
```

2. Quando já estiver conectado ao servidor, crie um banco de dados em branco no prompt.

```
CREATE DATABASE mypgsqldb;
```

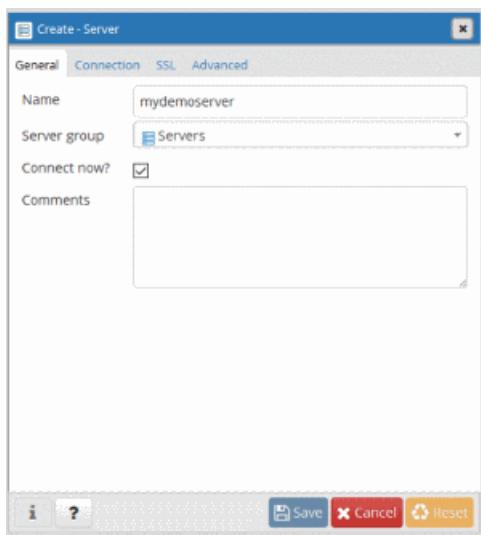
3. No prompt, execute o seguinte comando para mudar a conexão para o banco de dados `mypgsqldb` recém-criado:

```
\c mypgsqldb
```

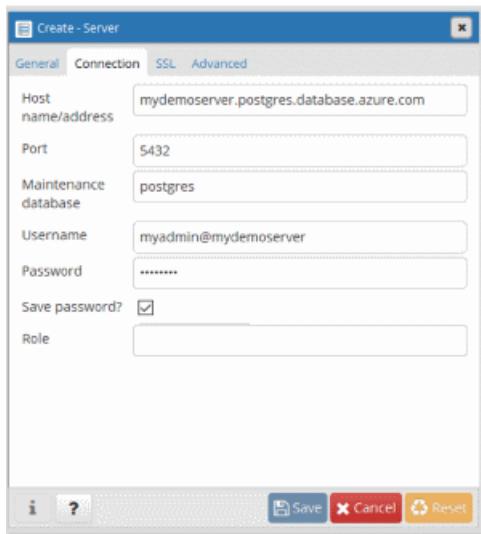
Conectar-se ao Servidor PostgreSQL usando pgAdmin

pgAdmin é uma ferramenta de software livre usada com PostgreSQL. Instale o pgAdmin por meio do [site do pgAdmin](#). A versão de pgAdmin que você está usando pode ser diferente da que é usada neste Início Rápido. Leia a documentação de pgAdmin se precisar de orientação adicional.

1. Abra o aplicativo pgAdmin no computador cliente.
2. Na barra de ferramentas, vá para **Objeto**, passe o mouse sobre **Criar** e selecione **Servidor**.
3. Na caixa de diálogo **Criar – Servidor**, na guia **Geral**, insira um nome amigável exclusivo para o servidor, como **mydemoserver**.



4. Na caixa de diálogo **Criar – Servidor** da guia **Conexão**, preencha a tabela de configurações.

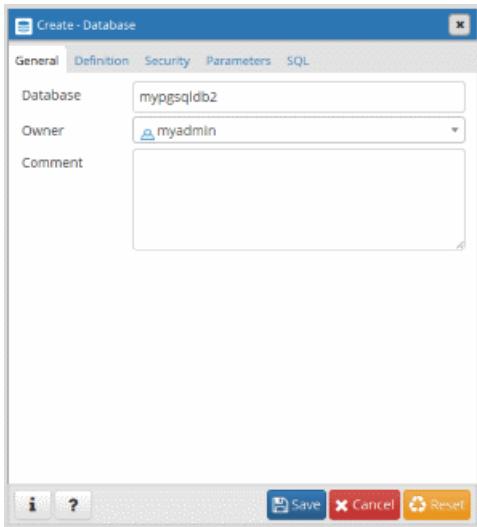


PARÂMETRO PGADMIN	VALOR	DESCRIÇÃO
-------------------	-------	-----------

PARÂMETRO PGADMIN	VALOR	DESCRIÇÃO
Nome/endereço do host	Nome do servidor	O valor do nome do servidor usado ao criar o Banco de Dados do Azure para o servidor PostgreSQL anteriormente. Nossa servidor de exemplo é mydemoserver.postgres.database.azure.com . Use o nome de domínio totalmente qualificado (*.postgres.database.azure.com), conforme mostrado no exemplo. Caso não se lembre do nome do servidor, siga as etapas da seção anterior para obter as informações de conexão.
Porta	5432	A porta a ser usada ao se conectar ao Banco de Dados do Azure para o servidor PostgreSQL.
Banco de dados de manutenção	<i>postgres</i>	O nome do banco de dados padrão gerado pelo sistema.
Nome de Usuário	Nome de logon do administrador do servidor	O nome de usuário de logon do administrador do servidor fornecido ao criar o Banco de Dados do Azure para o servidor PostgreSQL anteriormente. Caso não se lembre do nome de usuário, siga as etapas da seção anterior para obter as informações de conexão. O formato é <i>nome de usuário@nome do servidor</i> .
Senha	Sua senha de administrador	A senha que você escolheu ao criar o servidor anteriormente neste Guia de início rápido.
Função	Deixar em branco	Não é necessário fornecer um nome de função neste momento. Deixe o campo em branco.
Modo SSL	<i>Exigir</i>	Você pode definir o modo TLS/SSL na guia SSL do pgAdmin. Por padrão, todos os servidores do Banco de Dados do Azure para PostgreSQL são criados com a imposição de TLS ligada. Para desligar a imposição de TLS, confira Configurar a imposição de TLS .

5. Clique em **Salvar**.
6. No painel **Navegador** à esquerda, expanda o nó **Servidores**. Selecione o servidor, por exemplo, **mydemoserver**. Clique nele para se conectar a ele.
7. Expanda o nó do servidor e expanda **Bancos de Dados** nele. A lista deve conter o banco de dados *postgres* existente e outros bancos de dados criados. Você pode criar vários bancos de dados por servidor com o Banco de Dados do Azure para PostgreSQL.

8. Clique com o botão direito do mouse em **Bancos de Dados**, escolha o menu **Criar** e, em seguida, selecione **Banco de Dados**.
9. Digite um nome de banco de dados de sua escolha no campo **Banco de Dados**, como **mypgsqldb2**.
10. Selecione o **Proprietário** do banco de dados na caixa de listagem. Escolha o nome de logon do administrador do servidor, como o exemplo, **myadmin**.



11. Selecione **Salvar** para criar um novo banco de dados em branco.
12. No painel **Procurar**, veja o banco de dados criado na lista de bancos de dados com o nome do servidor.

Limpar os recursos

Se os recursos criados neste guia de início rápido não forem necessários para outro guia de início rápido ou tutorial, você poderá excluí-los executando o exemplo a seguir.

Caution

O exemplo a seguir exclui o grupo de recursos especificado e todos os recursos contidos nele. Se existirem recursos fora do escopo deste guia de início rápido no grupo de recursos especificado, eles também serão excluídos.

```
Remove-AzResourceGroup -Name myresourcegroup
```

Para excluir somente o servidor criado neste guia de início rápido sem excluir o grupo de recursos, use o cmdlet `Remove-AzPostgreSqlServer`.

```
Remove-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup
```

Próximas etapas

[Criar um Banco de Dados do Azure para PostgreSQL usando o PowerShell](#)

Início Rápido: Usar um modelo do ARM para criar um Banco de Dados do Azure para PostgreSQL – servidor único

01/07/2021 • 9 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço gerenciado usado para executar, gerenciar e dimensionar bancos de dados altamente disponíveis do PostgreSQL na nuvem. Neste guia de início rápido, você usa um modelo do ARM (modelo do Azure Resource Manager) para criar um Banco de Dados do Azure para PostgreSQL – servidor único no portal do Azure, no PowerShell ou na CLI do Azure.

Um [modelo ARM](#) é um arquivo JSON (JavaScript Object Notation) que define a infraestrutura e a configuração do projeto. O modelo usa a sintaxe declarativa. Na sintaxe declarativa, você descreve a implantação pretendida sem gravar a sequência de comandos de programação para criar a implantação.

Se seu ambiente atender aos pré-requisitos e você estiver familiarizado com o uso de modelos ARM, selecione o botão **Implantar no Azure**. O modelo será aberto no portal do Azure.



Pré-requisitos

- [Portal](#)
- [PowerShell](#)
- [CLI](#)

Uma conta do Azure com uma assinatura ativa. [Crie um gratuitamente](#).

Examinar o modelo

Você cria um Banco de Dados do Azure para PostgreSQL com um conjunto configurado de recursos de computação e armazenamento. Para saber mais, consulte [Tipos de preço no Banco de Dados do Azure para PostgreSQL – Servidor único](#). Crie o serviço dentro de um [Grupo de recursos do Azure](#).

O modelo usado neste início rápido é proveniente dos [Modelos de Início Rápido do Azure](#).

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "serverName": {  
            "type": "string",  
            "metadata": {  
                "description": "Server Name for Azure database for PostgreSQL"  
            }  
        },  
        "administratorLogin": {  
            "type": "string",  
            "minLength": 1,  
            "metadata": {  
                "description": "Database administrator login name"  
            }  
        },  
        "administratorLoginPassword": {  
            "type": "string",  
            "minLength": 1,  
            "maxLength": 128,  
            "metadata": {  
                "description": "Database administrator password"  
            }  
        }  
    },  
    "variables": {},  
    "resources": [  
        {  
            "type": "Microsoft.DBforPostgreSQL/servers",  
            "name": "[parameters('serverName')]",  
            "apiVersion": "2017-12-01",  
            "location": "West US",  
            "properties": {  
                "administratorLogin": "[parameters('administratorLogin')]",  
                "administratorLoginPassword": "[parameters('administratorLoginPassword')]",  
                "version": "12.0",  
                "sku": {  
                    "name": "B_Gen5_1",  
                    "tier": "Gen5",  
                    "family": "General Purpose",  
                    "capacity": 1  
                },  
                "tags": {  
                    "environment": "Development"  
                }  
            }  
        }  
    ]  
}
```

```
"administratorLoginPassword": {
    "type": "securestring",
    "minLength": 8,
    "metadata": {
        "description": "Database administrator password"
    }
},
"skuCapacity": {
    "type": "int",
    "defaultValue": 2,
    "metadata": {
        "description": "Azure database for PostgreSQL compute capacity in vCores (2,4,8,16,32)"
    }
},
"skuName": {
    "type": "string",
    "defaultValue": "GP_Gen5_2",
    "metadata": {
        "description": "Azure database for PostgreSQL sku name"
    }
},
"skuSizeMB": {
    "type": "int",
    "defaultValue": 51200,
    "metadata": {
        "description": "Azure database for PostgreSQL Sku Size"
    }
},
"skuTier": {
    "type": "string",
    "defaultValue": "GeneralPurpose",
    "metadata": {
        "description": "Azure database for PostgreSQL pricing tier"
    }
},
"skuFamily": {
    "type": "string",
    "defaultValue": "Gen5",
    "metadata": {
        "description": "Azure database for PostgreSQL sku family"
    }
},
"postgresqlVersion": {
    "type": "string",
    "defaultValue": "11",
    "allowedValues": [
        "9.5",
        "9.6",
        "10",
        "11"
    ],
    "metadata": {
        "description": "PostgreSQL version"
    }
},
"location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]",
    "metadata": {
        "description": "Location for all resources."
    }
},
"backupRetentionDays": {
    "type": "int",
    "defaultValue": 7,
    "metadata": {
        "description": "PostgreSQL Server backup retention days"
    }
},
```

```
"geoRedundantBackup": {
    "type": "string",
    "defaultValue": "Disabled",
    "metadata": {
        "description": "Geo-Redundant Backup setting"
    }
},
"virtualNetworkName": {
    "type": "string",
    "defaultValue": "azure_postgresql_vnet",
    "metadata": {
        "description": "Virtual Network Name"
    }
},
"subnetName": {
    "type": "string",
    "defaultValue": "azure_postgresql_subnet",
    "metadata": {
        "description": "Subnet Name"
    }
},
"virtualNetworkRuleName": {
    "type": "string",
    "defaultValue": "AllowSubnet",
    "metadata": {
        "description": "Virtual Network RuleName"
    }
},
"vnetAddressPrefix": {
    "type": "string",
    "defaultValue": "10.0.0.0/16",
    "metadata": {
        "description": "Virtual Network Address Prefix"
    }
},
"subnetPrefix": {
    "type": "string",
    "defaultValue": "10.0.0.0/16",
    "metadata": {
        "description": "Subnet Address Prefix"
    }
}
},
"variables": {
    "firewallrules": {
        "batch": {
            "rules": [
                {
                    "Name": "rule1",
                    "StartIpAddress": "0.0.0.0",
                    "EndIpAddress": "255.255.255.255"
                },
                {
                    "Name": "rule2",
                    "StartIpAddress": "0.0.0.0",
                    "EndIpAddress": "255.255.255.255"
                }
            ]
        }
    }
},
"resources": [
{
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2020-06-01",
    "name": "[parameters('virtualNetworkName')]",
    "location": "[parameters('location')]",
    "properties": {
        "addressSpace": {
```

```

        "addressPrefixes": [
            "[parameters('vnetAddressPrefix')]"
        ]
    },
    "resources": [
        {
            "type": "subnets",
            "apiVersion": "2020-06-01",
            "name": "[parameters('subnetName')]",
            "location": "[parameters('location')]",
            "dependsOn": [
                "[parameters('virtualNetworkName')]"
            ],
            "properties": {
                "addressPrefix": "[parameters('subnetPrefix')]"
            }
        }
    ]
},
{
    "type": "Microsoft.DBforPostgreSQL/servers",
    "apiVersion": "2017-12-01",
    "name": "[parameters('serverName')]",
    "location": "[parameters('location')]",
    "sku": {
        "name": "[parameters('skuName')]",
        "tier": "[parameters('skuTier')]",
        "capacity": "[parameters('skuCapacity')]",
        "size": "[parameters('skuSizeMB')]",
        "family": "[parameters('skuFamily')]"
    },
    "properties": {
        "createMode": "Default",
        "version": "[parameters('postgresqlVersion')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "[parameters('administratorLoginPassword')]",
        "storageProfile": {
            "storageMB": "[parameters('skuSizeMB')]",
            "backupRetentionDays": "[parameters('backupRetentionDays')]",
            "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
        }
    },
    "resources": [
        {
            "type": "virtualNetworkRules",
            "apiVersion": "2017-12-01",
            "name": "[parameters('virtualNetworkRuleName')]",
            "dependsOn": [
                "[resourceId('Microsoft.DBforPostgreSQL/servers/', parameters('serverName'))]"
            ],
            "properties": {
                "virtualNetworkSubnetId": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]",
                "ignoreMissingVnetServiceEndpoint": true
            }
        }
    ],
    {
        "type": "Microsoft.DBforPostgreSQL/servers/firewallRules",
        "apiVersion": "2017-12-01",
        "name": "
[concat(parameters('serverName'), '/', variables('firewallrules').batch.rules[copyIndex()].Name)]",
        "location": "[parameters('location')]",
        "copy": {
            "name": "firewallRulesCopy",
            "mode": "Serial",
            "batchSize": 1,

```

```
        "count": "[length(variables('firewallrules').batch.rules)]"
    },
    "dependsOn": [
        "[resourceId('Microsoft.DBforPostgreSQL/servers/', parameters('serverName'))]"
    ],
    "properties": {
        "startIpAddress": "[variables('firewallrules').batch.rules[copyIndex()].StartIpAddress]",
        "endIpAddress": "[variables('firewallrules').batch.rules[copyIndex()].EndIpAddress]"
    }
}
]
}
```

O modelo define cinco recursos do Azure:

- [Microsoft.Network/virtualNetworks](#)
- [Microsoft.Network/virtualNetworks/subnets](#)
- [Microsoft.DBforPostgreSQL/servers](#)
- [Microsoft.DBforPostgreSQL/servers/virtualNetworkRules](#)
- [Microsoft.DBforPostgreSQL/servers/firewallRules](#)

Mais exemplos de modelos do Banco de Dados do Azure para PostgreSQL podem ser encontrados nos [Modelos de Início Rápido do Azure](#).

Implantar o modelo

- [Portal](#)
- [PowerShell](#)
- [CLI](#)

Selecione o seguinte link para implantar o modelo de servidor do Banco de Dados do Azure para PostgreSQL no portal do Azure:



Na página [Implantar Banco de Dados do Azure para PostgreSQL com VNet](#):

1. Para **Grupo de recursos**, selecione **Criar**, insira um nome para o novo grupo de recursos e, em seguida, selecione **OK**.
2. Caso tenha criado um grupo de recursos, selecione uma **Localização** para o grupo de recursos e para o novo servidor.
3. Insira um **Nome do Servidor**, um **Logon de Administrador** e uma **Senha de Logon de Administrador**.

The screenshot shows the Azure portal interface for deploying a PostgreSQL database with VNet support. The top navigation bar includes the Azure logo, a search bar, and user account information. The main title is "Implantar o banco de dados do PostgreSQL com VNet".

MODELO

- Template: 101-managed-postgresql-with-vnet (3 recursos)
- Buttons: Editar modelo, Editar parâme..., Saiba mais

NOÇÕES BÁSICAS

- Assinatura *: Contoso
- Grupo de recursos *: Criar novo
- Localização *: (EUA) Oeste dos EUA 2

CONFIGURAÇÕES

- Nome do servidor *: (empty input field)
- Logon de administrador *: (empty input field)
- Senha de login do administrador... *: (empty input field)
- Capacidade de SKU: 2
- Nome do SKU: GP_Gen5_2
- Tamanho do SKU MB: 51200
- Nível de SKU: GeneralPurpose
- Família do SKU: Gen5
- Versão do Postgresql: 11
- Localização: [resourceGroup().location]
- Dias de retenção de backup: 7
- Backup de redundância geográfica: Desabilitado
- Nome da Rede Virtual: azure_postgresql_vnet
- Nome da Sub-rede: azure_postgresql_subnet
- Nome da regra da rede virtual: AllowSubnet
- Prefixo de Endereço da VNet: 10.0.0.0/16

ACTIONS

- Comprá (Buy)

4. Se desejar, altere as outras configurações padrão:

- **Assinatura:** a assinatura do Azure que você deseja usar para o servidor.
- **Capacidade de SKU:** a capacidade vCore, que pode ser 2 (o padrão), 4, 8, 16, 32 ou 64.
- **Nome do SKU:** o prefixo do nível de SKU, a família de SKU e a capacidade de SKU, unidos por sublinhados, como *B_Gen5_1*, *GP_Gen5_2* (o padrão) ou *MO_Gen5_32*.
- **Tamanho do SKU em MB:** o tamanho do armazenamento, em megabytes, do servidor do Banco de Dados do Azure para PostgreSQL (padrão 51200).
- **Nível do SKU:** o nível de implantação, como *Basic*, *GeneralPurpose* (o padrão) ou *MemoryOptimized*.
- **Família do SKU:** *Gen4* ou *Gen5* (o padrão), que indica a geração de hardware para a implantação do servidor.
- **Versão do PostgreSQL:** a versão do servidor PostgreSQL a ser implantada, como *9.5*, *9.6*, *10* ou *11* (a padrão).
- **Dias de retenção de backup:** o período desejado para retenção de backup com redundância geográfica, em dias (padrão 7).
- **Backup de redundância geográfica:** *Habilitado* ou *Desabilitado* (o padrão), dependendo dos

requisitos de Geo-DR (recuperação de desastre geográfico).

- **Nome da Rede Virtual:** o nome da rede virtual (padrão `azure_postgresql_vnet`).
- **Nome da Sub-Rede:** o nome da sub-rede (padrão `azure_postgresql_subnet`).
- **Nome da Regra da Rede Virtual:** o nome da regra da rede virtual que permite a sub-rede (padrão `AllowSubnet`).
- **Prefixo de Endereço da VNET:** o prefixo de endereço para a rede virtual (padrão `10.0.0.0/16`).
- **Prefixo da Sub-Rede:** o prefixo de endereço da sub-rede (padrão `10.0.0.0/16`).

5. Leia os termos e condições e depois selecione **Eu concordo com os termos e condições declarados acima.**

6. Selecione **Comprar**.

Examinar os recursos implantados

- [Portal](#)
- [PowerShell](#)
- [CLI](#)

Siga estas etapas para ter uma visão geral do seu novo servidor do Banco de Dados do Azure para PostgreSQL:

1. No [portal do Azure](#), pesquise e selecione **servidores do Banco de Dados do Azure para PostgreSQL**.
2. Na lista banco de dados, selecione o novo servidor. A página **Visão Geral** do novo servidor do Banco de Dados do Azure para PostgreSQL é exibida.

Como exportar um modelo do ARM por meio do portal

Você pode [exportar um modelo do ARM](#) pelo portal do Azure. Há duas maneiras de exportar um modelo:

- [Exportação do grupo de recursos ou do recurso](#). Essa opção gera um novo modelo com base nos recursos existentes. O modelo exportado é um "instantâneo" do estado atual do grupo de recursos. Você pode exportar um grupo de recursos inteiro ou recursos específicos dentro desse grupo de recursos.
- [Exportação antes da implantação ou do histórico](#). Essa opção recupera uma cópia exata de um modelo usado para implantação.

Quando você exportar o modelo, na seção `"properties": {}` do recurso de servidor PostgreSQL, você observará que `administratorLogin` e `administratorLoginPassword` não serão incluídos por motivos de segurança. Você **PRECISARÁ** adicionar esses parâmetros ao modelo antes de implantá-lo ou ele falhará.

```
"resources": [
  {
    "type": "Microsoft.DBforPostgreSQL/servers",
    "apiVersion": "2017-12-01",
    "name": "[parameters('servers_name')]",
    "location": "southcentralus",
    "sku": {
      "name": "B_Gen5_1",
      "tier": "Basic",
      "family": "Gen5",
      "capacity": 1
    },
    "properties": {
      "administratorLogin": "[parameters('administratorLogin')]",
      "administratorLoginPassword": "[parameters('administratorLoginPassword')]"
    }
  }
]
```

Limpar os recursos

Quando não for mais necessário, exclua o grupo de recursos, que excluirá os recursos no grupo de recursos.

- [Portal](#)
- [PowerShell](#)
- [CLI](#)

1. No [portal do Azure](#) pesquise e selecione **Grupos de recursos**.
2. Na lista grupo de recursos, escolha o nome do seu grupo de recursos.
3. Na página **Visão geral** do grupo de recursos, selecione **Excluir grupo de recursos**.
4. Na caixa de diálogo de confirmação, digite o nome do seu grupo de recursos e, em seguida, selecione **Excluir**.

Próximas etapas

Para obter um tutorial passo a passo que orienta você durante o processo de criação de um modelo, confira:

[Tutorial: Criar e implantar seu primeiro modelo do Resource Manager](#)

Início Rápido: Usar o Python para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 4 minutes to read

Neste início rápido, você aprenderá a se conectar ao Banco de Dados do Azure para PostgreSQL servidor único e executar instruções SQL para consultar usando o Python em macOS, Ubuntu Linux ou Windows.

TIP

Se você estiver procurando criar um aplicativo Django com PostgreSQL, confira o tutorial [Implantar um aplicativo Web Django com PostgreSQL](#).

Pré-requisitos

Para este início rápido você precisa:

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Criar um Banco de Dados do Azure para PostgreSQL servidor único usando o [portal do Azure](#) ou a [CLI do Azure](#) se ainda não tiver um.
- Com base em se você está usando o acesso público ou privado, conclua **UMA** das ações abaixo para habilitar a conectividade.

AÇÃO	MÉTODO DE CONECTIVIDADE	GUIA DE INSTRUÇÕES
Configurar regras de firewall	Público	Portal CLI
Configurar Ponto de Extremidade de Serviço	Público	Portal CLI
Configurar link privado	Privados	Portal CLI

- [Python](#) 2.7 ou 3.6+.
- Instalador do pacote [pip](#) mais recente.
- Instale o [psycopg2](#) usando `pip install psycopg2` em um terminal ou em uma janela do prompt de comando. Para obter mais informações, confira [como instalar psycopg2](#).

Obter informações da conexão de banco de dados

A conexão com um Banco de Dados do Azure para PostgreSQL requer o nome do servidor totalmente qualificado e as credenciais de logon. Você pode obter essas informações no portal do Azure.

1. No [portal do Azure](#), procure e selecione o nome do servidor do Banco de Dados do Azure para PostgreSQL.

2. Na página Visão Geral do servidor, copie o **Nome do servidor** totalmente qualificado e o **Nome de usuário do administrador**. O **Nome do servidor** totalmente qualificado sempre está no formato <my-server-name>.postgres.database.azure.com e o **Nome do usuário administrador** sempre está no formato <my-admin-username>@<my-server-name> .

Você também precisa da sua senha de administrador. Se a esquecer, será possível redefiní-la nessa página.

The screenshot shows the Microsoft Azure portal interface for managing a PostgreSQL server. The left sidebar shows navigation options like 'Visão geral', 'Log de atividades', 'Controle de Acesso (IAM)', 'Marcas', 'Diagnosticar e resolver pro...', 'Configurações', 'Segurança da conexão', and 'Cadeias de conexão'. The main content area is titled 'Microsoft Azure' and shows details for the server 'postgresv'. A red box highlights the 'Redefinir senha' (Change Password) button. Another red box highlights the 'Nome do servidor' (Server Name) field containing 'postgresv.postgres.database.azure.com' and the 'Nome de usuário administrador' (Administrator User Name) field containing 'azureuser@postgresv'.

IMPORTANT

Substitua os seguintes valores:

- <server-name> e <admin-username> pelos valores copiados do portal do Azure.
- <admin-password> pela sua senha de servidor.
- <database-name> um banco de dados padrão chamado *postgres* foi criado automaticamente quando você criou o servidor. É possível renomear esse banco de dados ou [criar outro](#) usando os comandos SQL.

Etapa 1: conectar e inserir dados

O exemplo de código a seguir se conecta ao Banco de Dados do Azure para PostgreSQL usando

- a função `psycopg2.connect` e carrega dados com uma instrução SQL **INSERT**.
- A função `cursor.execute` executa a consulta SQL no banco de dados.

```

import psycopg2

# Update connection string information
host = "<server-name>"
dbname = "<database-name>"
user = "<admin-username>"
password = "<admin-password>"
sslmode = "require"

# Construct connection string
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password,
sslmode)
conn = psycopg2.connect(conn_string)
print("Connection established")

cursor = conn.cursor()

# Drop previous table of same name if one exists
cursor.execute("DROP TABLE IF EXISTS inventory;")
print("Finished dropping table (if existed)")

# Create a table
cursor.execute("CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);")
print("Finished creating table")

# Insert some data into the table
cursor.execute("INSERT INTO inventory (name, quantity) VALUES (%s, %s);", ("banana", 150))
cursor.execute("INSERT INTO inventory (name, quantity) VALUES (%s, %s);", ("orange", 154))
cursor.execute("INSERT INTO inventory (name, quantity) VALUES (%s, %s);", ("apple", 100))
print("Inserted 3 rows of data")

# Clean up
conn.commit()
cursor.close()
conn.close()

```

Quando o código é executado com êxito, ele produz a seguinte saída:

```

C:\ Command Prompt
c:\>cd postgres
c:\postgres>python postgres.py
Connection established
Finished dropping table (if existed)
Finished creating table
Inserted 3 rows of data
c:\postgres>

```

[Está com problemas? Fale conosco](#)

Etapa 2: Ler dados

O exemplo de código a seguir se conecta ao Banco de Dados do Azure para PostgreSQL e usa

- a função `cursor.execute` com a instrução SQL `SELECT` para ler dados.
- A função `cursor.fetchall()` aceita uma consulta e retorna um conjunto de resultados a ser percorrido usando

```
# Fetch all rows from table
cursor.execute("SELECT * FROM inventory;")
rows = cursor.fetchall()

# Print all rows
for row in rows:
    print("Data row = (%s, %s, %s)" %(str(row[0]), str(row[1]), str(row[2])))
```

[Está com problemas? Fale conosco](#)

Etapa 3: Atualizar dados

O exemplo de código a seguir usa a função `cursor.execute` com a instrução SQL `UPDATE` para atualizar dados.

```
# Update a data row in the table
cursor.execute("UPDATE inventory SET quantity = %s WHERE name = %s;", (200, "banana"))
print("Updated 1 row of data")
```

[Está com problemas? Fale conosco](#)

Etapa 5: Excluir dados

O exemplo de código a seguir executa a função `cursor.execute` com a instrução SQL `DELETE` para excluir um item do estoque inserido anteriormente.

```
# Delete data row from table
cursor.execute("DELETE FROM inventory WHERE name = %s;", ("orange",))
print("Deleted 1 row of data")
```

[Está com problemas? Fale conosco](#)

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Próximas etapas

[Gerenciar o servidor de Banco de Dados do Azure para MySQL usando o portal](#)

[Gerenciar o servidor de Banco de Dados do Azure para MySQL usando a CLI](#)

[Não foi possível encontrar o que estava procurando? Fale conosco.](#)

Início Rápido: Usar o Node.js para conectar-se e consultar dados no Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 5 minutes to read

Neste guia de início rápido, você se conectará a um Banco de Dados do Azure para PostgreSQL usando um aplicativo Node.js. Ele mostra como usar instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados. As etapas neste artigo pressupõem que você está familiarizado com o desenvolvimento usando Node.js e que começou recentemente a trabalhar com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Conclusão do Início Rápido: Criar um servidor de Banco de Dados do Azure para PostgreSQL no portal do Azure ou Início Rápido: Criar um Banco de Dados do Azure para PostgreSQL usando a CLI do Azure.
- [Node.js](#)

Instalar o cliente pg

Instalar [pg](#), que é um cliente PostgreSQL para Node.js.

Para fazer isso, execute o npm (gerenciador de pacotes de nó) para JavaScript na linha de comando para instalar o cliente pg.

```
npm install pg
```

Verifique a instalação listando os pacotes instalados.

```
npm list
```

Obter informações de conexão

Obtenha as informações de conexão necessárias para se conectar ao Banco de Dados do Azure para PostgreSQL. Você precisa das credenciais de logon e do nome do servidor totalmente qualificado.

1. No [portal do Azure](#), pesquise e selecione o servidor que você criou (como **mydemoserver**).
2. No painel **Visão geral** do servidor, anote o **Nome do servidor** e **Nome do usuário administrador**. Se você esquecer sua senha, também poderá redefini-la nesse painel.

Executar o código JavaScript no Node.js

Você pode iniciar o Node.js do shell bash, do Terminal ou do prompt de comando do Windows digitando `node` e executar o exemplo de código JavaScript interativamente copiando-o e colando-o no prompt. Como alternativa, você pode salvar o código JavaScript em um arquivo de texto e iniciar `node filename.js` com o nome do arquivo como um parâmetro para executá-lo.

Conectar-se, criar tabela e inserir dados

Use o código a seguir para se conectar e carregar os dados usando instruções SQL `CREATE TABLE` e `INSERT INTO`. O objeto `pg.Client` é usado para se comunicar com o servidor PostgreSQL. A função `pg.Client.connect()` é usada para estabelecer conexão com o servidor. A função `pg.Client.query()` é usada para executar a consulta SQL no banco de dados PostgreSQL.

Substitua os parâmetros `host`, `dbname`, `user` e `password` pelos valores que você especificou ao criar o servidor e o banco de dados.

```

const pg = require('pg');

const config = {
  host: '<your-db-server-name>.postgres.database.azure.com',
  // Do not hard code your username and password.
  // Consider using Node environment variables.
  user: '<your-db-username>',
  password: '<your-password>',
  database: '<name-of-database>',
  port: 5432,
  ssl: true
};

const client = new pg.Client(config);

client.connect(err => {
  if (err) throw err;
  else {
    queryDatabase();
  }
});

function queryDatabase() {
  const query = `
    DROP TABLE IF EXISTS inventory;
    CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);
    INSERT INTO inventory (name, quantity) VALUES ('banana', 150);
    INSERT INTO inventory (name, quantity) VALUES ('orange', 154);
    INSERT INTO inventory (name, quantity) VALUES ('apple', 100);
  `;

  client
    .query(query)
    .then(() => {
      console.log('Table created successfully!');
      client.end(console.log('Closed client connection'));
    })
    .catch(err => console.log(err))
    .then(() => {
      console.log('Finished execution, exiting now');
      process.exit();
    });
}
}

```

Ler dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL `SELECT`. O objeto `pg.Client` é usado para se comunicar com o servidor PostgreSQL. A função `pg.Client.Connect()` é usada para estabelecer conexão com o servidor. A função `pg.Client.Query()` é usada para executar a consulta SQL no banco de dados PostgreSQL.

Substitua os parâmetros `host`, `dbname`, `user` e `password` pelos valores que você especificou ao criar o servidor e o banco de dados.

```

const pg = require('pg');

const config = {
  host: '<your-db-server-name>.postgres.database.azure.com',
  // Do not hard code your username and password.
  // Consider using Node environment variables.
  user: '<your-db-username>',
  password: '<your-password>',
  database: '<name-of-database>',
  port: 5432,
  ssl: true
};

const client = new pg.Client(config);

client.connect(err => {
  if (err) throw err;
  else { queryDatabase(); }
});

function queryDatabase() {

  console.log(`Running query to PostgreSQL server: ${config.host}`);

  const query = 'SELECT * FROM inventory';

  client.query(query)
    .then(res => {
      const rows = res.rows;

      rows.map(row => {
        console.log(`Read: ${JSON.stringify(row)}`);
      });

      process.exit();
    })
    .catch(err => {
      console.log(err);
    });
}

```

Atualizar dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL **UPDATE**. O objeto [pg.Client](#) é usado para se comunicar com o servidor PostgreSQL. A função [pg.Client.Connect\(\)](#) é usada para estabelecer conexão com o servidor. A função [pg.Client.Query\(\)](#) é usada para executar a consulta SQL no banco de dados PostgreSQL.

Substitua os parâmetros host, dbname, user e password pelos valores que você especificou ao criar o servidor e o banco de dados.

```

const pg = require('pg');

const config = {
    host: '<your-db-server-name>.postgres.database.azure.com',
    // Do not hard code your username and password.
    // Consider using Node environment variables.
    user: '<your-db-username>',
    password: '<your-password>',
    database: '<name-of-database>',
    port: 5432,
    ssl: true
};

const client = new pg.Client(config);

client.connect(err => {
    if (err) throw err;
    else {
        queryDatabase();
    }
});

function queryDatabase() {
    const query = `
        UPDATE inventory
        SET quantity= 1000 WHERE name='banana';
    `;

    client
        .query(query)
        .then(result => {
            console.log('Update completed');
            console.log(`Rows affected: ${result.rowCount}`);
        })
        .catch(err => {
            console.log(err);
            throw err;
        });
}

```

Excluir dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL `DELETE`. O objeto `pg.Client` é usado para se comunicar com o servidor PostgreSQL. A função `pg.Client.Connect()` é usada para estabelecer conexão com o servidor. A função `pg.Client.Query()` é usada para executar a consulta SQL no banco de dados PostgreSQL.

Substitua os parâmetros `host`, `dbname`, `user` e `password` pelos valores que você especificou ao criar o servidor e o banco de dados.

```

const pg = require('pg');

const config = {
    host: '<your-db-server-name>.postgres.database.azure.com',
    // Do not hard code your username and password.
    // Consider using Node environment variables.
    user: '<your-db-username>',
    password: '<your-password>',
    database: '<name-of-database>',
    port: 5432,
    ssl: true
};

const client = new pg.Client(config);

client.connect(err => {
    if (err) {
        throw err;
    } else {
        queryDatabase();
    }
});

function queryDatabase() {
    const query = `
        DELETE FROM inventory
        WHERE name = 'apple';
    `;

    client
        .query(query)
        .then(result => {
            console.log('Delete completed');
            console.log(`Rows affected: ${result.rowCount}`);
        })
        .catch(err => {
            console.log(err);
            throw err;
        });
}
}

```

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```

az group delete \
--name $AZ_RESOURCE_GROUP \
--yes

```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

Início Rápido: Usar o Java e o JDBC com o Banco de Dados do Azure para PostgreSQL

12/08/2021 • 10 minutes to read

Este tópico demonstra como criar um aplicativo de exemplo que usa o Java e o [JDBC](#) para armazenar e recuperar informações no [Banco de Dados do Azure para PostgreSQL](#).

O JDBC é a API Java padrão para se conectar a bancos de dados relacionais tradicionais.

Pré-requisitos

- Uma conta do Azure. Se você não tiver uma, [obtenha uma avaliação gratuita](#).
- [Azure Cloud Shell](#) ou [CLI do Azure](#). É recomendável usar o Azure Cloud Shell para que o logon seja feito automaticamente e você tenha acesso a todas as ferramentas necessárias.
- Um [Java Development Kit](#) compatível, versão 8 (incluído no Azure Cloud Shell).
- A ferramenta de build [Apache Maven](#).

Preparar o ambiente de trabalho

Vamos usar as variáveis de ambiente para limitar erros de digitação e facilitar a personalização da configuração a seguir para as suas necessidades específicas.

Configure essas variáveis de ambiente usando os seguintes comandos:

```
AZ_RESOURCE_GROUP=database-workshop
AZ_DATABASE_NAME=<YOUR_DATABASE_NAME>
AZ_LOCATION=<YOUR_AZURE_REGION>
AZ_POSTGRESQL_USERNAME=demo
AZ_POSTGRESQL_PASSWORD=<YOUR_POSTGRESQL_PASSWORD>
AZ_LOCAL_IP_ADDRESS=<YOUR_LOCAL_IP_ADDRESS>
```

Substitua os espaços reservados pelos seguintes valores, que são usados em todo este artigo:

- <YOUR_DATABASE_NAME> : O nome do servidor PostgreSQL. O nome deve ser exclusivo em todo o Azure.
- <YOUR_AZURE_REGION> : A região do Azure que você usará. Você pode usar `eastus` por padrão, mas é recomendável configurar uma região mais próxima de onde você mora. Você pode ter a lista completa de regiões disponíveis ao digitar `az account list-locations`.
- <YOUR_POSTGRESQL_PASSWORD> : A senha do servidor de banco de dados PostgreSQL. Essa senha deveria ter um mínimo de oito caracteres. Os caracteres deveriam ser de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números (0-9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
- <YOUR_LOCAL_IP_ADDRESS> : O endereço IP do computador local, no qual você executará o aplicativo Java. Uma forma conveniente de encontrá-lo é apontar o navegador para whatismyip.akamai.com.

Em seguida, crie um grupo de recursos usando o seguinte comando:

```
az group create \
--name $AZ_RESOURCE_GROUP \
--location $AZ_LOCATION \
| jq
```

NOTE

Usamos o utilitário `jq` para exibir os dados JSON e torná-los mais legíveis. Esse utilitário é instalado por padrão no [Azure Cloud Shell](#). Se você não gostar desse utilitário, poderá removê-lo com segurança a parte `| jq` de todos os comandos que usaremos.

Criar uma instância do Banco de Dados do Azure para PostgreSQL

A primeira coisa que criaremos é um servidor PostgreSQL gerenciado.

NOTE

Você pode ler informações mais detalhadas sobre como criar servidores PostgreSQL em [Criar um servidor do Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#).

No [Azure Cloud Shell](#), execute o seguinte comando:

```
az postgres server create \
  --resource-group $AZ_RESOURCE_GROUP \
  --name $AZ_DATABASE_NAME \
  --location $AZ_LOCATION \
  --sku-name B_Gen5_1 \
  --storage-size 5120 \
  --admin-user $AZ_POSTGRESQL_USERNAME \
  --admin-password $AZ_POSTGRESQL_PASSWORD \
| jq
```

Esse comando cria um servidor PostgreSQL pequeno.

Configurar uma regra de firewall para o servidor PostgreSQL

As instâncias do Banco de Dados do Azure para PostgreSQL são protegidas por padrão. Elas têm um firewall que não permite nenhuma conexão de entrada. Para usar o banco de dados, você precisa adicionar uma regra de firewall que permitirá que o endereço IP local acesse o servidor de banco de dados.

Como você configurou seu endereço IP local no início deste artigo, abra o firewall do servidor executando o seguinte comando:

```
az postgres server firewall-rule create \
  --resource-group $AZ_RESOURCE_GROUP \
  --name $AZ_DATABASE_NAME--database-allow-local-ip \
  --server $AZ_DATABASE_NAME \
  --start-ip-address $AZ_LOCAL_IP_ADDRESS \
  --end-ip-address $AZ_LOCAL_IP_ADDRESS \
| jq
```

Configurar um banco de dados PostgreSQL

O servidor PostgreSQL que você criou anteriormente está vazio. Ele não tem nenhum banco de dados que você possa usar com o aplicativo Java. Crie um banco de dados chamado `demo` usando o seguinte comando:

```
az postgres db create \
  --resource-group $AZ_RESOURCE_GROUP \
  --name demo \
  --server-name $AZ_DATABASE_NAME \
| jq
```

Criar um projeto Java

Usando o seu IDE favorito, crie um projeto Java e adicione um arquivo `pom.xml` no diretório raiz:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>

  <properties>
    <java.version>1.8</java.version>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.2.12</version>
    </dependency>
  </dependencies>
</project>
```

Esse arquivo é um [Apache Maven](#) que configura o projeto a ser usado:

- Java 8
- Um driver PostgreSQL recente para Java

Preparar um arquivo de configuração para se conectar ao Banco de Dados do Azure para PostgreSQL

Crie o arquivo `src/main/resources/application.properties` e adicione:

```
url=jdbc:postgresql://$AZ_DATABASE_NAME.postgres.database.azure.com:5432/demo?ssl=true&sslmode=require
user=demo@$AZ_DATABASE_NAME
password=$AZ_POSTGRESQL_PASSWORD
```

- Substitua as duas variáveis `$AZ_DATABASE_NAME` pelo valor que você configurou no início deste artigo.
- Substitua a variável `$AZ_POSTGRESQL_PASSWORD` pelo valor que você configurou no início deste artigo.

NOTE

Acrescentamos `?ssl=true&sslmode=require` à propriedade de configuração `url` para instruir o driver JDBC a usar o [protocolo TLS](#) ao se conectar ao banco de dados. O uso do TLS com o Banco de Dados do Azure para PostgreSQL é obrigatório e uma boa prática de segurança.

Criar um arquivo SQL para gerar o esquema de banco de dados

Usaremos um arquivo `src/main/resources/schema.sql` para criar um esquema de banco de dados. Crie esse arquivo com o seguinte conteúdo:

```
DROP TABLE IF EXISTS todo;
CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255), details VARCHAR(4096), done BOOLEAN);
```

Codificar o aplicativo

Conectarse ao banco de dados

Em seguida, adicione o código Java que usará o JDBC para armazenar e recuperar dados do servidor PostgreSQL.

Crie um arquivo `src/main/java/DemoApplication.java` que contém:

```
package com.example.demo;

import java.sql.*;
import java.util.*;
import java.util.logging.Logger;

public class DemoApplication {

    private static final Logger log;

    static {
        System.setProperty("java.util.logging.SimpleFormatter.format", "[%4$-7s] %5$s %n");
        log =Logger.getLogger(DemoApplication.class.getName());
    }

    public static void main(String[] args) throws Exception {
        log.info("Loading application properties");
        Properties properties = new Properties();

        properties.load(DemoApplication.class.getClassLoader().getResourceAsStream("application.properties"));

        log.info("Connecting to the database");
        Connection connection = DriverManager.getConnection(properties.getProperty("url"), properties);
        log.info("Database connection test: " + connection.getCatalog());

        log.info("Create database schema");
        Scanner scanner = new
Scanner(DemoApplication.class.getClassLoader().getResourceAsStream("schema.sql"));
        Statement statement = connection.createStatement();
        while (scanner.hasNextLine()) {
            statement.execute(scanner.nextLine());
        }

        /*
        Todo todo = new Todo(1L, "configuration", "congratulations, you have set up JDBC correctly!", true);
        insertData(todo, connection);
        todo = readData(connection);
        todo.setDetails("congratulations, you have updated data!");
        updateData(todo, connection);
        deleteData(todo, connection);
        */

        log.info("Closing database connection");
        connection.close();
    }
}
```

Esse código Java usará os arquivos `application.properties` e `schema.sql` que criamos anteriormente para se conectar ao servidor PostgreSQL e criar um esquema que armazenará nossos dados.

Nesse arquivo, você pode ver que comentamos os métodos para inserir, ler, atualizar e excluir dados: codificaremos esses métodos no restante deste artigo e você poderá remover as marcas de comentários uma após a outra.

NOTE

As credenciais de banco de dados são armazenadas nas propriedades *user* e *password* no arquivo *application.properties*. Essas credenciais são usadas durante a execução de `DriverManager.getConnection(properties.getProperty("url"), properties);`, pois o arquivo de propriedades é passado como um argumento.

Agora, você pode executar esta classe principal com a sua ferramenta favorita:

- Usando o IDE, você deve clicar com o botão direito do mouse na classe *DemoApplication* e executá-la.
- Usando o Maven, você pode executar o aplicativo por meio do:

```
mvn exec:java -Dexec.mainClass="com.example.demo.DemoApplication".
```

O aplicativo deve se conectar ao Banco de Dados do Azure para PostgreSQL, criar um esquema de banco de dados e fechar a conexão, como você vê nos logs do console:

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Closing database connection
```

Criar uma classe de domínio

Crie uma classe Java `Todo`, ao lado da classe `DemoApplication` e adicione o seguinte código:

```

package com.example.demo;

public class Todo {

    private Long id;
    private String description;
    private String details;
    private boolean done;

    public Todo() {
    }

    public Todo(Long id, String description, String details, boolean done) {
        this.id = id;
        this.description = description;
        this.details = details;
        this.done = done;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getDetails() {
        return details;
    }

    public void setDetails(String details) {
        this.details = details;
    }

    public boolean isDone() {
        return done;
    }

    public void setDone(boolean done) {
        this.done = done;
    }

    @Override
    public String toString() {
        return "Todo{" +
            "id=" + id +
            ", description='" + description + '\'' +
            ", details='" + details + '\'' +
            ", done=" + done +
            '}';
    }
}

```

Essa classe é um modelo de domínio mapeado na tabela `todo` que você criou ao executar o script `schema.sql`.

Inserir dados no Banco de Dados do Azure para PostgreSQL

No arquivo `src/main/java/DemoApplication.java`, após o método principal, adicione o seguinte método para

inserir dados no banco de dados:

```
private static void insertData(Todo todo, Connection connection) throws SQLException {
    log.info("Insert data");
    PreparedStatement insertStatement = connection
        .prepareStatement("INSERT INTO todo (id, description, details, done) VALUES (?, ?, ?, ?);");

    insertStatement.setLong(1, todo.getId());
    insertStatement.setString(2, todo.getDescription());
    insertStatement.setString(3, todo.getDetails());
    insertStatement.setBoolean(4, todo.isDone());
    insertStatement.executeUpdate();
}
```

Agora você pode remover a marca de comentário das seguintes duas linhas no método `main`:

```
Todo todo = new Todo(1L, "configuration", "congratulations, you have set up JDBC correctly!", true);
insertData(todo, connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO] Loading application properties
[INFO] Connecting to the database
[INFO] Database connection test: demo
[INFO] Create database schema
[INFO] Insert data
[INFO] Closing database connection
```

Como ler dados do Banco de Dados do Azure para PostgreSQL

Vamos ler os dados inseridos anteriormente para validar se o nosso código funciona corretamente.

No arquivo `src/main/java/DemoApplication.java`, após o método `insertData`, adicione o seguinte método para ler dados do banco de dados:

```
private static Todo readData(Connection connection) throws SQLException {
    log.info("Read data");
    PreparedStatement readStatement = connection.prepareStatement("SELECT * FROM todo;");
    ResultSet resultSet = readStatement.executeQuery();
    if (!resultSet.next()) {
        log.info("There is no data in the database!");
        return null;
    }
    Todo todo = new Todo();
    todo.setId(resultSet.getLong("id"));
    todo.setDescription(resultSet.getString("description"));
    todo.setDetails(resultSet.getString("details"));
    todo.setDone(resultSet.getBoolean("done"));
    log.info("Data read from the database: " + todo.toString());
    return todo;
}
```

Agora você pode remover a marca de comentário da seguinte linha no método `main`:

```
todo = readData(connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have set up JDBC correctly!', done=true}
[INFO    ] Closing database connection
```

Como atualizar dados no Banco de Dados do Azure para PostgreSQL

Vamos atualizar os dados que inserimos anteriormente.

Ainda no arquivo `src/main/java/DemoApplication.java`, após o método `readData`, adicione o seguinte método para atualizar os dados no banco de dados:

```
private static void updateData(Todo todo, Connection connection) throws SQLException {
    log.info("Update data");
    PreparedStatement updateStatement = connection
        .prepareStatement("UPDATE todo SET description = ?, details = ?, done = ? WHERE id = ?");

    updateStatement.setString(1, todo.getDescription());
    updateStatement.setString(2, todo.getDetails());
    updateStatement.setBoolean(3, todo.isDone());
    updateStatement.setLong(4, todo.getId());
    updateStatement.executeUpdate();
    readData(connection);
}
```

Agora você pode remover a marca de comentário das seguintes duas linhas no método `main`:

```
todo.setDetails("congratulations, you have updated data!");
updateData(todo, connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have set up JDBC correctly!', done=true}
[INFO    ] Update data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have updated data!', done=true}
[INFO    ] Closing database connection
```

Como excluir dados no Banco de Dados do Azure para PostgreSQL

Por fim, vamos excluir os dados que inserimos anteriormente.

Ainda no arquivo `src/main/java/DemoApplication.java`, após o método `updateData`, adicione o seguinte método para excluir os dados no banco de dados:

```
private static void deleteData(Todo todo, Connection connection) throws SQLException {
    log.info("Delete data");
    PreparedStatement deleteStatement = connection.prepareStatement("DELETE FROM todo WHERE id = ?;");
    deleteStatement.setLong(1, todo.getId());
    deleteStatement.executeUpdate();
    readData(connection);
}
```

Agora você pode remover a marca de comentário da seguinte linha no método `main`:

```
deleteData(todo, connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO ] Loading application properties
[INFO ] Connecting to the database
[INFO ] Database connection test: demo
[INFO ] Create database schema
[INFO ] Insert data
[INFO ] Read data
[INFO ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have set up JDBC correctly!', done=true}
[INFO ] Update data
[INFO ] Read data
[INFO ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have updated data!', done=true}
[INFO ] Delete data
[INFO ] Read data
[INFO ] There is no data in the database!
[INFO ] Closing database connection
```

Limpar os recursos

Parabéns! Você criou um aplicativo Java que usa o JDBC para armazenar e recuperar dados do Banco de Dados do Azure para PostgreSQL.

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

Início Rápido: Usar o Ruby para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 5 minutes to read

Este guia de início rápido demonstra como se conectar a um banco de dados do Azure para PostgreSQL usando aplicativo [Ruby](#). Ele mostra como usar instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados. As etapas neste artigo pressupõem que você esteja familiarizado com o desenvolvimento usando Ruby e que tenha começado a trabalhar recentemente com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

Este guia de início rápido usa os recursos criados em um destes guias como ponto de partida:

- [Criar BD – Portal](#)
- [Criar Banco de dados - CLI do Azure](#)

Você também precisa ter instalados:

- [Ruby](#)
- [Ruby pg](#), o módulo PostgreSQL para Ruby

Obter informações de conexão

Obtenha as informações de conexão necessárias para se conectar ao Banco de Dados do Azure para PostgreSQL. Você precisa das credenciais de logon e do nome do servidor totalmente qualificado.

1. Faça logon no [Portal do Azure](#).
2. No menu à esquerda no portal do Azure, clique em **Todos os recursos** e pesquise o servidor que você criou (como **mydemoserver**).
3. Clique no nome do servidor.
4. No painel **Visão Geral** do servidor, anote o **Nome do servidor** e **Nome de logon do administrador do servidor**. Se você esquecer sua senha, também poderá redefini-la nesse painel.

The screenshot shows the Azure portal interface. On the left, there's a sidebar with icons for storage, databases, and other services. The main area shows a list of resources under 'Todos os recursos'. A specific resource named 'mydemoserver' is selected and highlighted with a red box. To the right, the 'Visão geral' (General) blade for this resource is displayed. It includes sections for 'Conceitos básicos' (Basic concepts) and 'CONFIGURAÇÕES' (Settings). In the 'Conceitos básicos' section, the 'Nome do servidor' (Server name) is listed as 'mydemoserver.postgres.database.azure.com' and the 'Nome de logon do administrador do servidor' (Administrator logon name) is listed as 'myadmin@mydemoserver'. Both of these lines are also highlighted with red boxes. At the bottom of the blade, there are links for 'Redefinir senha' (Reset password), 'Restaurar' (Restore), and 'Excluir' (Delete).

NOTE

O símbolo `@` no nome de usuário Postgres do Azure tem sido codificado por URL como `%40` em todas as cadeias de conexão.

Conectarse e crear una tabla

Use o código a seguir para se conectar e criar uma tabela usando a instrução SQL `CREATE TABLE`, seguida por instruções SQL `INSERT INTO` para adicionar linhas à tabela.

O código usa um objeto `PG::Connection` com o construtor `new` para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método `exec()` para executar os comandos `DROP`, `CREATE TABLE` e `INSERT INTO`. O código verifica erros usando a classe `PG::Error`. Em seguida, ele chama o método `close()` para fechar a conexão antes de encerrar. Consulte a documentação de referência do Ruby PG para obter mais informações sobre essas classes e métodos.

Substitua as cadeias de caracteres `host`, `database`, `user` e `password` pelos seus próprios valores.

```
require 'pg'

begin
  # Initialize connection variables.
  host = String('mydemoserver.postgres.database.azure.com')
  database = String('postgres')
  user = String('mylogin%40mydemoserver')
  password = String('<server_admin_password>')

  # Initialize connection object.
  connection = PG::Connection.new(:host => host, :user => user, :dbname => database, :port => '5432',
  :password => password)
  puts 'Successfully created connection to database'

  # Drop previous table of same name if one exists
  connection.exec('DROP TABLE IF EXISTS inventory;')
  puts 'Finished dropping table (if existed).'

  # Create new table.
  connection.exec('CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);')
  puts 'Finished creating table.'

  # Insert some data into table.
  connection.exec("INSERT INTO inventory VALUES(1, 'banana', 150)")
  connection.exec("INSERT INTO inventory VALUES(2, 'orange', 154)")
  connection.exec("INSERT INTO inventory VALUES(3, 'apple', 100)")
  puts 'Inserted 3 rows of data.'

rescue PG::Error => e
  puts e.message

ensure
  connection.close if connection
end
```

Ler dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL `SELECT`.

O código usa um objeto `PG::Connection` com construtor `new` para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método `exec()` para executar o comando `SELECT`, mantendo os resultados em um conjunto de resultados. A coleção do conjunto de resultados é iterada usando o loop `resultSet.each do`, mantendo os valores da linha atual na variável `row`. O código verifica erros usando a classe `PG::Error`. Em seguida, ele chama o método `close()` para fechar a conexão antes de encerrar. Consulte a documentação de referência do Ruby PG para obter mais informações sobre essas classes e métodos.

Substitua as cadeias de caracteres `host`, `database`, `user` e `password` pelos seus próprios valores.

```

require 'pg'

begin
    # Initialize connection variables.
    host = String('mydemoserver.postgres.database.azure.com')
    database = String('postgres')
    user = String('mylogin%40mydemoserver')
    password = String('<server_admin_password>')

    # Initialize connection object.
    connection = PG::Connection.new(:host => host, :user => user, :database => dbname, :port => '5432',
:password => password)
    puts 'Successfully created connection to database.'

    resultSet = connection.exec('SELECT * from inventory;')
    resultSet.each do |row|
        puts 'Data row = (%s, %s, %s)' % [row['id'], row['name'], row['quantity']]
    end

rescue PG::Error => e
    puts e.message

ensure
    connection.close if connection
end

```

Atualizar dados

Use o código a seguir para conectar-se e atualizar os dados usando uma instrução SQL UPDATE.

O código usa um objeto `PG::Connection` com construtor `new` para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método `exec()` para executar o comando UPDATE. O código verifica erros usando a classe `PG::Error`. Em seguida, ele chama o método `close()` para fechar a conexão antes de encerrar. Consulte a documentação de referência do [Ruby PG](#) para obter mais informações sobre essas classes e métodos.

Substitua as cadeias de caracteres `host`, `database`, `user` e `password` pelos seus próprios valores.

```

require 'pg'

begin
    # Initialize connection variables.
    host = String('mydemoserver.postgres.database.azure.com')
    database = String('postgres')
    user = String('mylogin%40mydemoserver')
    password = String('<server_admin_password>')

    # Initialize connection object.
    connection = PG::Connection.new(:host => host, :user => user, :dbname => database, :port => '5432',
:password => password)
    puts 'Successfully created connection to database.'

    # Modify some data in table.
    connection.exec('UPDATE inventory SET quantity = %d WHERE name = %s;' % [200, '\'banana\''])
    puts 'Updated 1 row of data.'

rescue PG::Error => e
    puts e.message

ensure
    connection.close if connection
end

```

Excluir dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL **DELETE**.

O código usa um objeto `PG::Connection` com construtor `new` para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método `exec()` para executar o comando UPDATE. O código verifica erros usando a classe `PG::Error`. Em seguida, ele chama o método `close()` para fechar a conexão antes de encerrar.

Substitua as cadeias de caracteres `host`, `database`, `user` e `password` pelos seus próprios valores.

```
require 'pg'

begin
    # Initialize connection variables.
    host = String('mydemoserver.postgres.database.azure.com')
    database = String('postgres')
    user = String('mylogin%40mydemoserver')
    password = String('<server_admin_password>')

    # Initialize connection object.
    connection = PG::Connection.new(:host => host, :user => user, :dbname => database, :port => '5432',
:password => password)
    puts 'Successfully created connection to database.'

    # Modify some data in table.
    connection.exec('DELETE FROM inventory WHERE name = %s;' % ['\'orange\''])
    puts 'Deleted 1 row of data.'

rescue PG::Error => e
    puts e.message

ensure
    connection.close if connection
end
```

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

[Documentação de referência do Ruby Pg](#)

Início Rápido: Usar o PHP para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 6 minutes to read

Este guia de início rápido demonstra como se conectar a um banco de dados do Azure para PostgreSQL usando aplicativo [PHP](#). Ele mostra como usar instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados. As etapas neste artigo pressupõem que você esteja familiarizado com o desenvolvimento usando PHP e que tenha começado a trabalhar recentemente com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

Este guia de início rápido usa os recursos criados em um destes guias como ponto de partida:

- [Criar BD – Portal](#)
- [Criar Banco de dados - CLI do Azure](#)

Instalar o PHP

Instalar o PHP em seu próprio servidor ou crie um [aplicativo Web](#) do Azure que inclua o PHP.

Windows

- Baixar o [PHP 7.1.4 versão protegida não thread \(x64\)](#)
- Instalar o PHP e consultar o [manual do PHP](#) para outras configurações
- O código usa a classe **pgsql** (`ext/php_pgsql.dll`) que está incluída na instalação do PHP.
- Habilitou a extensão **pgsql** editando o arquivo de configuração `php.ini`, geralmente localizado em `C:\Program Files\PHP\v7.1\php.ini`. O arquivo de configuração deve conter uma linha com o texto `extension=php_pgsql.so`. Se não for exibido, adicione o texto e salve o arquivo. Se o texto estiver presente, mas comentado com um prefixo de ponto e vírgula, remova a marca de comentário do texto removendo o ponto e vírgula.

Linux (Ubuntu)

- Baixar o [PHP 7.1.4 versão protegida não thread \(x64\)](#)
- Instalar o PHP e consultar o [manual do PHP](#) para outras configurações
- O código usa a classe **pgsql** (`php_pgsql.so`). Instale-a executando `sudo apt-get install php-pgsql`.
- Habilitou a extensão **pgsql** editando o arquivo de configuração `/etc/php/7.0/mods-available/pgsql.ini`. O arquivo de configuração deve conter uma linha com o texto `extension=php_pgsql.so`. Se não for exibido, adicione o texto e salve o arquivo. Se o texto estiver presente, mas comentado com um prefixo de ponto e vírgula, remova a marca de comentário do texto removendo o ponto e vírgula.

MacOS

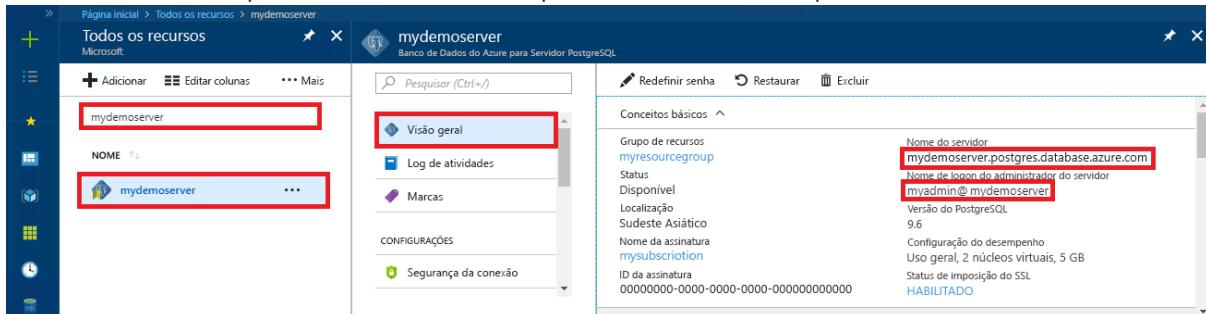
- Baixar o [PHP 7.1.4 versão](#)
- Instalar o PHP e consultar o [manual do PHP](#) para outras configurações

Obter informações de conexão

Obtenha as informações de conexão necessárias para se conectar ao Banco de Dados do Azure para

PostgreSQL. Você precisa das credenciais de logon e do nome do servidor totalmente qualificado.

1. Faça logon no [Portal do Azure](#).
2. No menu à esquerda no portal do Azure, clique em **Todos os recursos** e pesquise o servidor que você criou (como **mydemoserver**).
3. Clique no nome do servidor.
4. No painel **Visão Geral** do servidor, anote o **Nome do servidor** e **Nome de logon do administrador do servidor**. Se você esquecer sua senha, também poderá redefini-la nesse painel.



The screenshot shows the Azure portal interface. On the left, there's a sidebar with icons for storage, databases, and more. The main area shows 'Todos os recursos' (All resources) with a search bar and a list of resources. One resource, 'mydemoserver', is selected and highlighted with a red box. To the right, there's a detailed view of the 'mydemoserver' resource. The 'Visão geral' (General View) tab is selected and highlighted with a red box. Under 'Conceitos básicos', the 'Nome do servidor' (Server name) is listed as 'mydemoserver.postgres.database.azure.com' and the 'Nome de logon do administrador do servidor' (Administrator logon name) is listed as 'myadmin@mydemoserver', both of which are also highlighted with red boxes.

Conectar-se e criar uma tabela

Use o código a seguir para se conectar e criar uma tabela usando a instrução SQL **CREATE TABLE**, seguida por instruções SQL **INSERT INTO** para adicionar linhas à tabela.

O método de chamada de código [pg_connect\(\)](#) para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método [pg_query\(\)](#) várias vezes para executar vários comandos e [pg_last_error\(\)](#) para verificar os detalhes em caso de erro. Em seguida, ele chama o método [pg_close\(\)](#) para fechar a conexão.

Substitua os parâmetros `$host`, `$database`, `$user`, e `$password` pelos seus próprios valores.

```

<?php

    // Initialize connection variables.
    $host = "mydemoserver.postgres.database.azure.com";
    $database = "mypgsqlldb";
    $user = "mylogin@mydemoserver";
    $password = "<server_admin_password>";

    // Initialize connection object.
    $connection = pg_connect("host=$host dbname=$database user=$user password=$password")
        or die("Failed to create connection to database: ". pg_last_error(). "<br/>");
    print "Successfully created connection to database.<br/>";

    // Drop previous table of same name if one exists.
    $query = "DROP TABLE IF EXISTS inventory;";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). "<br/>");
    print "Finished dropping table (if existed).<br/>";

    // Create table.
    $query = "CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). "<br/>");
    print "Finished creating table.<br/>";

    // Insert some data into table.
    $name = '\'banana\'';
    $quantity = 150;
    $query = "INSERT INTO inventory (name, quantity) VALUES ($name, $quantity);";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). "<br/>");

    $name = '\'orange\'';
    $quantity = 154;
    $query = "INSERT INTO inventory (name, quantity) VALUES ($name, $quantity);";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). "<br/>");

    $name = '\'apple\'';
    $quantity = 100;
    $query = "INSERT INTO inventory (name, quantity) VALUES ($name, $quantity);";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). "<br/>");

    print "Inserted 3 rows of data.<br/>";

    // Closing connection
    pg_close($connection);
?>

```

Ler dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL SELECT.

O método de chamada de código [pg_connect\(\)](#) para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método [pg_query\(\)](#) para executar o comando SELECT, mantendo os resultados em um conjunto de resultados, e [pg_last_error\(\)](#) para verificar os detalhes em caso de erro. Para ler o resultado definido, o método [pg_fetch_row\(\)](#) é chamado em um loop, uma vez por linha, e a linha de dados é recuperada de uma matriz `$row`, com um valor de dados por coluna em cada posição de matriz. Para liberar o resultado definido, o método [pg_free_result\(\)](#) é chamado. Em seguida, ele chama o método [pg_close\(\)](#) para fechar a conexão.

Substitua os parâmetros `$host`, `$database`, `$user`, e `$password` pelos seus próprios valores.

```

<?php
    // Initialize connection variables.
    $host = "mydemoserver.postgres.database.azure.com";
    $database = "mypgsqlldb";
    $user = "mylogin@mydemoserver";
    $password = "<server_admin_password>";

    // Initialize connection object.
    $connection = pg_connect("host=$host dbname=$database user=$user password=$password")
        or die("Failed to create connection to database: ". pg_last_error(). "<br/>");

    print "Successfully created connection to database. <br/>";

    // Perform some SQL queries over the connection.
    $query = "SELECT * from inventory";
    $result_set = pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). "<br/>");
    while ($row = pg_fetch_row($result_set))
    {
        print "Data row = ($row[0], $row[1], $row[2]). <br/>";
    }

    // Free result_set
    pg_free_result($result_set);

    // Closing connection
    pg_close($connection);
?>

```

Atualizar dados

Use o código a seguir para conectar-se e atualizar os dados usando uma instrução SQL **UPDATE**.

O método de chamada de código [pg_connect\(\)](#) para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método [pg_query\(\)](#) para executar um comando e [pg_last_error\(\)](#) para verificar os detalhes em caso de erro. Em seguida, ele chama o método [pg_close\(\)](#) para fechar a conexão.

Substitua os parâmetros `$host`, `$database`, `$user`, e `$password` pelos seus próprios valores.

```

<?php
    // Initialize connection variables.
    $host = "mydemoserver.postgres.database.azure.com";
    $database = "mypgsqlldb";
    $user = "mylogin@mydemoserver";
    $password = "<server_admin_password>";

    // Initialize connection object.
    $connection = pg_connect("host=$host dbname=$database user=$user password=$password")
        or die("Failed to create connection to database: ". pg_last_error(). ".<br/>");

    print "Successfully created connection to database. <br/>";

    // Modify some data in table.
    $new_quantity = 200;
    $name = '\'banana\'';
    $query = "UPDATE inventory SET quantity = $new_quantity WHERE name = $name;";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). ".<br/>");

    print "Updated 1 row of data. <br/>";

    // Closing connection
    pg_close($connection);
?>

```

Excluir dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL **DELETE**.

O método de chamada de código [pg_connect\(\)](#) para se conectar ao Banco de Dados do Azure para PostgreSQL. Em seguida, ele chama o método [pg_query\(\)](#) para executar um comando e [pg_last_error\(\)](#) para verificar os detalhes em caso de erro. Em seguida, ele chama o método [pg_close\(\)](#) para fechar a conexão.

Substitua os parâmetros `$host`, `$database`, `$user`, e `$password` pelos seus próprios valores.

```
<?php
    // Initialize connection variables.
    $host = "mydemoserver.postgres.database.azure.com";
    $database = "mypgsqldb";
    $user = "mylogin@mydemoserver";
    $password = "<server_admin_password>";

    // Initialize connection object.
    $connection = pg_connect("host=$host dbname=$database user=$user password=$password")
        or die("Failed to create connection to database: ". pg_last_error(). ". <br>");

    print "Successfully created connection to database. <br/>";

    // Delete some data from table.
    $name = '\'orange\'';
    $query = "DELETE FROM inventory WHERE name = $name;";
    pg_query($connection, $query)
        or die("Encountered an error when executing given sql statement: ". pg_last_error(). ". <br/>");
    print "Deleted 1 row of data. <br/>";

    // Closing connection
    pg_close($connection);
?>
```

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

Início Rápido: Usar o .NET (C#) para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 6 minutes to read

Este guia de início rápido demonstra como se conectar a um banco de dados do Azure para PostgreSQL usando aplicativo C#. Ele mostra como usar instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados. As etapas neste artigo pressupõem que você está familiarizado com o desenvolvimento usando C# e que começou recentemente a trabalhar com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

Para este início rápido você precisa:

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Criar um Banco de Dados do Azure para PostgreSQL servidor único usando o [portal do Azure](#) ou a [CLI do Azure](#) se ainda não tiver um.
- Com base em se você está usando o acesso público ou privado, conclua **UMA** das ações abaixo para habilitar a conectividade.

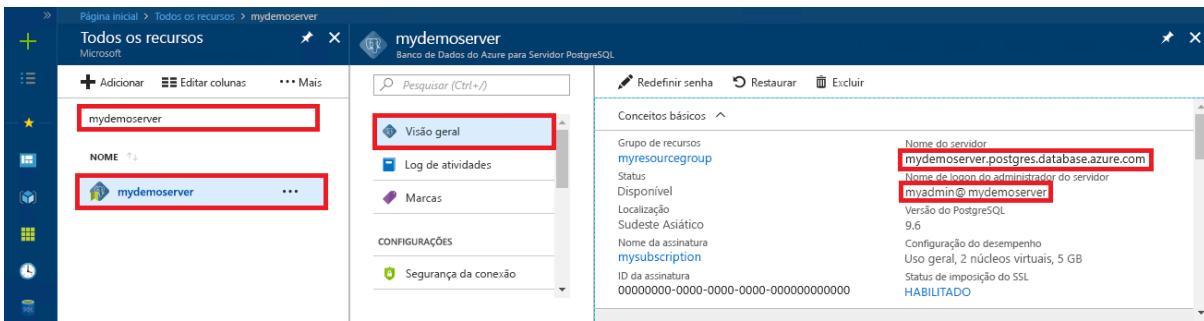
AÇÃO	MÉTODO DE CONECTIVIDADE	GUIA DE INSTRUÇÕES
Configurar regras de firewall	Público	Portal CLI
Configurar Ponto de Extremidade de Serviço	Público	Portal CLI
Configurar link privado	Privados	Portal CLI

- Instale o [.NET Framework](#) para sua plataforma (Windows, Ubuntu Linux ou macOS).
- Instale o [Visual Studio](#) para criar seu projeto.
- Instale o pacote NuGet [Npgsql](#) no Visual Studio.

Obter informações de conexão

Obtenha as informações de conexão necessárias para se conectar ao Banco de Dados do Azure para PostgreSQL. Você precisa das credenciais de logon e do nome do servidor totalmente qualificado.

1. Faça logon no [Portal do Azure](#).
2. No menu à esquerda no portal do Azure, clique em **Todos os recursos** e pesquise o servidor que você criou (como **mydemoserver**).
3. Clique no nome do servidor.
4. No painel **Visão Geral** do servidor, anote o **Nome do servidor** e **Nome de logon do administrador do servidor**. Se você esquecer sua senha, também poderá redefini-la nesse painel.



Etapa 1: conectar e inserir dados

Use o código a seguir para se conectar e carregar os dados usando instruções SQL **CREATE TABLE** e **INSERT INTO**. O código usa a classe `NpgsqlCommand` com o método:

- `Open()` para estabelecer uma conexão com o banco de dados PostgreSQL.
- `CreateCommand()` para definir a propriedade `CommandText`.
- `ExecuteNonQuery()` para executar os comandos de banco de dados.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```
using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresCreate
    {
        // Obtain connection string information from the portal
        //

        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin@mydemoserver";
        private static string DBname = "mypgsqldb";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //

            string connString =
                String.Format(
                    "Server={0};Username={1};Database={2};Port={3};Password={4};SSLMode=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))

            {
                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("DROP TABLE IF EXISTS inventory", conn))
                {
                    command.ExecuteNonQuery();
                }
            }
        }
    }
}
```

```
        Console.Out.WriteLine("Finished dropping table (if existed)");

    }

    using (var command = new NpgsqlCommand("CREATE TABLE inventory(id serial PRIMARY KEY, name
VARCHAR(50), quantity INTEGER)", conn))
    {
        command.ExecuteNonQuery();
        Console.Out.WriteLine("Finished creating table");
    }

    using (var command = new NpgsqlCommand("INSERT INTO inventory (name, quantity) VALUES (@n1,
@q1), (@n2, @q2), (@n3, @q3)", conn))
    {
        command.Parameters.AddWithValue("n1", "banana");
        command.Parameters.AddWithValue("q1", 150);
        command.Parameters.AddWithValue("n2", "orange");
        command.Parameters.AddWithValue("q2", 154);
        command.Parameters.AddWithValue("n3", "apple");
        command.Parameters.AddWithValue("q3", 100);

        int nRows = command.ExecuteNonQuery();
        Console.Out.WriteLine(String.Format("Number of rows inserted={0}", nRows));
    }
}

Console.WriteLine("Press RETURN to exit");
Console.ReadLine();
}
}
```

Está com problemas? Fale conosco.

Etapa 2: Ler dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL SELECT. O código usa a classe NpgsqlCommand com o método:

- `Open()` para estabelecer uma conexão com o PostgreSQL.
 - `CreateCommand()` e `ExecuteReader()` para executar os comandos de banco de dados.
 - `Read()` a fim de avançar para os registros nos resultados.
 - `GetInt32()` e `GetString()` para analisar os valores do registro.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```

using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresRead
    {
        // Obtain connection string information from the portal
        //
        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin@mydemoserver";
        private static string DBname = "mypgsqlldb";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //
            string connString =
                String.Format(
                    "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSlMode=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))
            {

                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("SELECT * FROM inventory", conn))
                {

                    var reader = command.ExecuteReader();
                    while (reader.Read())
                    {
                        Console.WriteLine(
                            string.Format(
                                "Reading from table={({0}, {1}, {2})}",
                                reader.GetInt32(0).ToString(),
                                reader.GetString(1),
                                reader.GetInt32(2).ToString()
                            )
                        );
                    }
                    reader.Close();
                }
            }

            Console.WriteLine("Press RETURN to exit");
            Console.ReadLine();
        }
    }
}

```

[Está com problemas? Fale conosco.](#)

Etapa 3: Atualizar dados

Use o código a seguir para conectar-se e atualizar os dados usando uma instrução SQL **UPDATE**. O código usa a

classe NpgsqlCommand com o método:

- [Open\(\)](#) para estabelecer uma conexão com o PostgreSQL.
- [CreateCommand\(\)](#) para definir a propriedade CommandText.
- [ExecuteNonQuery\(\)](#) para executar os comandos de banco de dados.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```
using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresUpdate
    {
        // Obtain connection string information from the portal
        //

        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin@mydemoserver";
        private static string DBname = "mypgsqlldb";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //

            string connString =
                String.Format(
                    "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSLMODE=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))
            {

                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("UPDATE inventory SET quantity = @q WHERE name = @n",
                conn))
                {
                    command.Parameters.AddWithValue("n", "banana");
                    command.Parameters.AddWithValue("q", 200);
                    int nRows = command.ExecuteNonQuery();
                    Console.Out.WriteLine(String.Format("Number of rows updated={0}", nRows));
                }
            }

            Console.WriteLine("Press RETURN to exit");
            Console.ReadLine();
        }
    }
}
```

Está com problemas? Fale conosco.

Etapa 4: Excluir dados

Use o código a seguir para conectar-se e excluir os dados usando uma instrução SQL **DELETE**.

O código usa a classe `NpgsqlCommand` com o método `Open()` para estabelecer uma conexão com o banco de dados PostgreSQL. Em seguida, o código usa o método `CreateCommand()`, define a propriedade `CommandText` e chama o método `ExecuteNonQuery ()` para executar os comandos do banco de dados.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```

using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresDelete
    {
        // Obtain connection string information from the portal
        //

        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin@mydemoserver";
        private static string DBname = "mypostgresql";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //
            string connString =
                String.Format(
                    "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSlMode=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))
            {
                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("DELETE FROM inventory WHERE name = @n", conn))
                {
                    command.Parameters.AddWithValue("n", "orange");

                    int nRows = command.ExecuteNonQuery();
                    Console.Out.WriteLine(String.Format("Number of rows deleted={0}", nRows));
                }
            }

            Console.WriteLine("Press RETURN to exit");
            Console.ReadLine();
        }
    }
}

```

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```

az group delete \
--name $AZ_RESOURCE_GROUP \
--yes

```

Próximas etapas

[Gerenciar o servidor de Banco de Dados do Azure para MySQL usando o portal](#)

[Gerenciar o servidor de Banco de Dados do Azure para MySQL usando a CLI](#)

Não foi possível encontrar o que estava procurando? Fale conosco.

Início Rápido: Usar a linguagem Go para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 9 minutes to read

Este guia de início rápido demonstra como se conectar a um banco de dados do Azure para PostgreSQL usando código escrito na linguagem [Go](#) (golang). Ele mostra como usar instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados. Este artigo pressupõem que você está familiarizado com o desenvolvimento usando Go, mas que começou recentemente a trabalhar com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

Este guia de início rápido usa os recursos criados em um destes guias como ponto de partida:

- [Criar BD – Portal](#)
- [Criar Banco de dados - CLI do Azure](#)

Instalar o conector pq e o Go

Instale o [Go](#) e o [driver de Postgres Go puro\(pq\)](#) em seu próprio computador. Dependendo da sua plataforma, siga as etapas apropriadas:

Windows

1. [Baixe](#) e instale o Go para Microsoft Windows de acordo com as [instruções de instalação](#).
2. Inicie o prompt de comando no menu Iniciar.
3. Crie uma pasta para o seu projeto, como `mkdir %USERPROFILE%\go\src\postgresqlgo`.
4. Altere o diretório na pasta do projeto, como `cd %USERPROFILE%\go\src\postgresqlgo`.
5. Defina a variável de ambiente para GOPATH apontar para o diretório de código de origem.
`set GOPATH=%USERPROFILE%\go`.
6. Instale o [driver de Postgres Go puro \(pq\)](#) executando o comando `go get github.com/lib/pq`.

Em resumo, instale o Go e execute esses comandos no prompt de comando:

```
mkdir %USERPROFILE%\go\src\postgresqlgo
cd %USERPROFILE%\go\src\postgresqlgo
set GOPATH=%USERPROFILE%\go
go get github.com/lib/pq
```

Linux (Ubuntu)

1. Abra o shell do Bash.
2. Instale o Go executando `sudo apt-get install golang-go`.
3. Crie uma pasta para o seu projeto em seu diretório inicial, como `mkdir -p ~/go/src/postgresqlgo/`.
4. Altere o diretório na pasta, como `cd ~/go/src/postgresqlgo/`.

- Defina a variável de ambiente GOPATH para apontar para um diretório de origem válido, como a pasta atual inicial do diretório do Go. No shell bash, execute `export GOPATH=~/go` para adicionar o diretório do Go como GOPATH para a sessão atual do shell.
- Instale o [driver de Postgres Go puro \(pq\)](#) executando o comando `go get github.com/lib/pq`.

Em resumo, execute estes comandos bash:

```
sudo apt-get install golang-go
mkdir -p ~/go/src/postgresqlgo/
cd ~/go/src/postgresqlgo/
export GOPATH=~/go/
go get github.com/lib/pq
```

Apple macOS

- Baixe e instale o Go de acordo com as [instruções de instalação](#) correspondentes à plataforma.
- Abra o shell do Bash.
- Crie uma pasta para o seu projeto em seu diretório inicial, como `mkdir -p ~/go/src/postgresqlgo/`.
- Altere o diretório na pasta, como `cd ~/go/src/postgresqlgo/`.
- Defina a variável de ambiente GOPATH para apontar para um diretório de origem válido, como a pasta atual inicial do diretório do Go. No shell bash, execute `export GOPATH=~/go` para adicionar o diretório do Go como GOPATH para a sessão atual do shell.
- Instale o [driver de Postgres Go puro \(pq\)](#) executando o comando `go get github.com/lib/pq`.

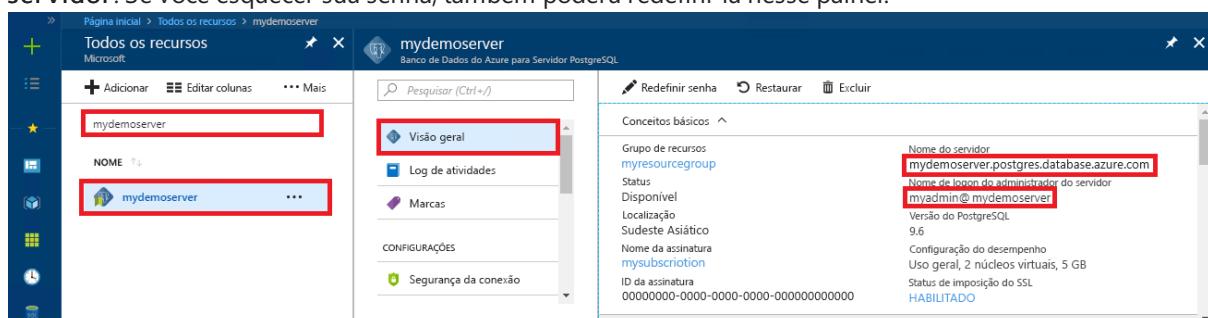
Em resumo, instale o Go e, em seguida, execute esses comandos bash:

```
mkdir -p ~/go/src/postgresqlgo/
cd ~/go/src/postgresqlgo/
export GOPATH=~/go/
go get github.com/lib/pq
```

Obter informações de conexão

Obtenha as informações de conexão necessárias para se conectar ao Banco de Dados do Azure para PostgreSQL. Você precisa das credenciais de logon e do nome do servidor totalmente qualificado.

- Faça logon no [Portal do Azure](#).
- No menu à esquerda no portal do Azure, clique em **Todos os recursos** e pesquise o servidor que você criou (como **mydemoserver**).
- Clique no nome do servidor.
- No painel **Visão Geral** do servidor, anote o **Nome do servidor** e **Nome de logon do administrador do servidor**. Se você esquecer sua senha, também poderá redefiní-la nesse painel.



Compilar e executar o código Go

1. Para escrever código Golang, você pode usar um editor de texto sem formatação, como o Bloco de Notas no Microsoft Windows, [vi](#) ou [Nano](#) no Ubuntu ou TextEdit no macOS. Se você preferir um IDE (Ambiente de Desenvolvimento Integrado) mais avançado, experimente o [GoLand](#) da Jetbrains, o [Visual Studio Code](#) da Microsoft ou o [Atom](#).
2. Cole o código Golang das seções a seguir em arquivos de texto e salve-os em sua pasta de projeto com a extensão de arquivo *.go, como caminho do Windows `%USERPROFILE%\go\src\postgresqlgo\createtable.go` ou caminho do Linux `~/go/src/postgresqlgo/createtable.go`.
3. Localize as constantes `HOST`, `DATABASE`, `USER`, e `PASSWORD` no código e substitua os valores de exemplo com seus próprios valores.
4. Inicie o prompt de comando ou shell Bash. Altere o diretório na pasta do seu projeto. Por exemplo, no Windows `cd %USERPROFILE%\go\src\postgresqlgo\`. No Linux `cd ~/go/src/postgresqlgo/`. Alguns dos ambientes IDE mencionados oferecem recursos de depuração e runtime sem a necessidade de comandos do shell.
5. Execute o código, digitando o comando `go run createtable.go` para compilar o aplicativo e executá-lo.
6. Como alternativa, para compilar o código em um aplicativo nativo, `go build createtable.go`, inicie `createtable.exe` para executar o aplicativo.

Conectar-se e criar uma tabela

Use o código a seguir para se conectar e criar uma tabela usando a instrução SQL `CREATE TABLE`, seguida por instruções SQL `INSERT INTO` para adicionar linhas à tabela.

O código importa três pacotes: o [pacote sql](#), o [pacote pq](#) como um driver para se comunicar com o servidor PostgreSQL e o [pacote fmt](#) para entrada e saída impressa na linha de comando.

O código chama o método `sql.Open()` para se conectar ao Banco de Dados do Azure para PostgreSQL e verifica a conexão usando o método `db.Ping()`. Um [identificador de banco de dados](#) é usado em todo o processo, mantendo o pool de conexão para o servidor de banco de dados. O código chama o método `Exec()` várias vezes para executar vários comandos SQL. Sempre que um método personalizado `checkError()` verificar se ocorreu um erro e for necessário sair em caso de erro.

Substitua os parâmetros `HOST`, `DATABASE`, `USER`, e `PASSWORD` pelos seus próprios valores.

```

package main

import (
    "database/sql"
    "fmt"
    _ "github.com/lib/pq"
)

const (
    // Initialize connection constants.
    HOST      = "mydemoserver.postgres.database.azure.com"
    DATABASE  = "mypgsqldb"
    USER      = "mylogin@mydemoserver"
    PASSWORD  = "<server_admin_password>"
)

func checkError(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {
    // Initialize connection string.
    var connectionString string = fmt.Sprintf("host=%s user=%s password=%s dbname=%s sslmode=require", HOST,
    USER, PASSWORD, DATABASE)

    // Initialize connection object.
    db, err := sql.Open("postgres", connectionString)
    checkError(err)

    err = db.Ping()
    checkError(err)
    fmt.Println("Successfully created connection to database")

    // Drop previous table of same name if one exists.
    _, err = db.Exec("DROP TABLE IF EXISTS inventory;")
    checkError(err)
    fmt.Println("Finished dropping table (if existed)")

    // Create table.
    _, err = db.Exec("CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);")
    checkError(err)
    fmt.Println("Finished creating table")

    // Insert some data into table.
    sql_statement := "INSERT INTO inventory (name, quantity) VALUES ($1, $2);"
    _, err = db.Exec(sql_statement, "banana", 150)
    checkError(err)
    _, err = db.Exec(sql_statement, "orange", 154)
    checkError(err)
    _, err = db.Exec(sql_statement, "apple", 100)
    checkError(err)
    fmt.Println("Inserted 3 rows of data")
}

```

Ler dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL SELECT.

O código importa três pacotes: o [pacote sql](#), o [pacote pq](#) como um driver para se comunicar com o servidor PostgreSQL e o [pacote fmt](#) para entrada e saída impressa na linha de comando.

O código chama o método [sql.Open\(\)](#) para se conectar ao Banco de Dados do Azure para PostgreSQL e verifica a conexão usando o método [db.Ping\(\)](#). Um [identificador de banco de dados](#) é usado em todo o processo,

mantendo o pool de conexão para o servidor de banco de dados. A consulta select é executada chamando o método `db.Query()`, e as linhas resultantes são mantidas em uma variável do tipo `linhas`. O código lê a coluna de valores de dados na linha atual usando o método `rows.Scan()` e loops em relação às linhas usando o iterador `rows.Next()` até que não haja mais linhas. Os valores de coluna de cada linha são impressos no console de saída. A cada ocorrência, um método personalizado `checkError()` é usado para verificar se ocorreu um erro e se é necessário sair em caso de erro.

Substitua os parâmetros `HOST`, `DATABASE`, `USER`, e `PASSWORD` pelos seus próprios valores.

```

package main

import (
    "database/sql"
    "fmt"
    _ "github.com/lib/pq"
)

const (
    // Initialize connection constants.
    HOST      = "mydemoserver.postgres.database.azure.com"
    DATABASE  = "mypgsqldb"
    USER      = "mylogin@mydemoserver"
    PASSWORD  = "<server_admin_password>"
)

func checkError(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {

    // Initialize connection string.
    var connectionString string = fmt.Sprintf("host=%s user=%s password=%s dbname=%s sslmode=require", HOST,
    USER, PASSWORD, DATABASE)

    // Initialize connection object.
    db, err := sql.Open("postgres", connectionString)
    checkError(err)

    err = db.Ping()
    checkError(err)
    fmt.Println("Successfully created connection to database")

    // Read rows from table.
    var id int
    var name string
    var quantity int

    sql_statement := "SELECT * from inventory;"
    rows, err := db.Query(sql_statement)
    checkError(err)
    defer rows.Close()

    for rows.Next() {
        switch err := rows.Scan(&id, &name, &quantity); err {
        case sql.ErrNoRows:
            fmt.Println("No rows were returned")
        case nil:
            fmt.Printf("Data row = (%d, %s, %d)\n", id, name, quantity)
        default:
            checkError(err)
        }
    }
}

```

Atualizar dados

Use o código a seguir para conectar-se e atualizar os dados usando uma instrução SQL UPDATE.

O código importa três pacotes: o [pacote sql](#), o [pacote pq](#) como driver para se comunicar com o servidor Postgres e o [fmt pacote](#) para impressão de entrada e saída na linha de comando.

O código chama o método `sql.Open()` para se conectar ao Banco de Dados do Azure para PostgreSQL e verifica a conexão usando o método `db.Ping()`. Um [identificador de banco de dados](#) é usado em todo o processo, mantendo o pool de conexão para o servidor de banco de dados. O código chama o método `Exec()` para executar a instrução SQL que atualiza a tabela. Um método personalizado `checkError()` é usado para verificar se ocorreu um erro e se é necessário sair em caso de erro.

Substitua os parâmetros `HOST`, `DATABASE`, `USER`, e `PASSWORD` pelos seus próprios valores.

```
package main

import (
    "database/sql"
    _ "github.com/lib/pq"
    "fmt"
)

const (
    // Initialize connection constants.
    HOST      = "mydemoserver.postgres.database.azure.com"
    DATABASE  = "mypgsqlldb"
    USER      = "mylogin@mydemoserver"
    PASSWORD  = "<server_admin_password>"
)

func checkError(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {

    // Initialize connection string.
    var connectionString string =
        fmt.Sprintf("host=%s user=%s password=%s dbname=%s sslmode=require", HOST, USER, PASSWORD, DATABASE)

    // Initialize connection object.
    db, err := sql.Open("postgres", connectionString)
    checkError(err)

    err = db.Ping()
    checkError(err)
    fmt.Println("Successfully created connection to database")

    // Modify some data in table.
    sql_statement := "UPDATE inventory SET quantity = $2 WHERE name = $1;"
    _, err = db.Exec(sql_statement, "banana", 200)
    checkError(err)
    fmt.Println("Updated 1 row of data")
}
```

Excluir dados

Use o código a seguir para conectar-se e excluir os dados usando uma instrução SQL `DELETE`.

O código importa três pacotes: o [pacote sql](#), o [pacote pq](#) como driver para se comunicar com o servidor Postgres e o [fmt pacote](#) para impressão de entrada e saída na linha de comando.

O código chama o método `sql.Open()` para se conectar ao Banco de Dados do Azure para PostgreSQL e verifica a conexão usando o método `db.Ping()`. Um [identificador de banco de dados](#) é usado em todo o processo, mantendo o pool de conexão para o servidor de banco de dados. O código chama o método `Exec()` para executar a instrução SQL que exclui uma linha da tabela. Um método personalizado `checkError()` é usado para

verificar se ocorreu um erro e se é necessário sair em caso de erro.

Substitua os parâmetros `HOST`, `DATABASE`, `USER`, e `PASSWORD` pelos seus próprios valores.

```
package main

import (
    "database/sql"
    _ "github.com/lib/pq"
    "fmt"
)

const (
    // Initialize connection constants.
    HOST      = "mydemoserver.postgres.database.azure.com"
    DATABASE  = "mypgsqldb"
    USER      = "mylogin@mydemoserver"
    PASSWORD  = "<server_admin_password>"
)

func checkError(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {

    // Initialize connection string.
    var connectionString string =
        fmt.Sprintf("host=%s user=%s password=%s dbname=%s sslmode=require", HOST, USER, PASSWORD, DATABASE)

    // Initialize connection object.
    db, err := sql.Open("postgres", connectionString)
    checkError(err)

    err = db.Ping()
    checkError(err)
    fmt.Println("Successfully created connection to database")

    // Delete some data from table.
    sql_statement := "DELETE FROM inventory WHERE name = $1;"
    _, err = db.Exec(sql_statement, "orange")
    checkError(err)
    fmt.Println("Deleted 1 row of data")
}
```

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
    --name $AZ_RESOURCE_GROUP \
    --yes
```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

Início Rápido: usar o Rust para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – Servidor Único

21/05/2021 • 7 minutes to read

Neste artigo você aprenderá a usar o [driver do PostgreSQL para Rust](#) a fim de interagir com o Banco de Dados do Azure para PostgreSQL explorando as operações CRUD (criar, ler, atualizar e excluir) implementadas no código de exemplo. Por fim, você pode executar o aplicativo localmente para vê-lo em ação.

Pré-requisitos

Para este início rápido você precisa:

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Uma versão recente do [Rust](#) instalada.
- Um Banco de Dados do Azure para PostgreSQL, servidor único – Crie um usando o [portal do Azure](#) ou a [CLI do Azure](#).
- Com base em se você está usando o acesso público ou privado, conclua UMA das ações abaixo para habilitar a conectividade.

AÇÃO	MÉTODO DE CONECTIVIDADE	GUIA DE INSTRUÇÕES
Configurar regras de firewall	Público	Portal CLI
Configurar Ponto de Extremidade de Serviço	Público	Portal CLI
Configurar link privado	Privados	Portal CLI

- [Git](#) instalado.

Obter informações da conexão de banco de dados

A conexão com um Banco de Dados do Azure para PostgreSQL requer o nome do servidor totalmente qualificado e as credenciais de logon. Você pode obter essas informações no portal do Azure.

1. No [portal do Azure](#), procure e selecione o nome do servidor do Banco de Dados do Azure para PostgreSQL.
2. Na página Visão Geral do servidor, copie o **Nome do servidor** totalmente qualificado e o **Nome de usuário do administrador**. O **Nome do servidor** totalmente qualificado sempre está no formato `<my-server-name>.postgres.database.azure.com` e **Nome do usuário administrador** sempre está no formato `<my-admin-username>@<my-server-name>`.

Examinar o código (opcional)

Se você estiver interessado em saber como o código funciona, poderá examinar os trechos de código a seguir.

Caso contrário, fique à vontade para pular para [Executar o aplicativo](#).

Conectar

A função `main` começa conectando-se ao Banco de Dados do Azure para PostgreSQL e depende das variáveis de ambiente a seguir para obter as informações de conectividade `POSTGRES_HOST`, `POSTGRES_USER`, `POSTGRES_PASSWORD` e `POSTGRES_DBNAME`. Por padrão, o serviço de banco de dados do PostgreSQL é configurado para exigir conexão `TLS`. Você pode optar por desabilitar a necessidade de `TLS` se o aplicativo cliente não oferecer suporte à conectividade `TLS`. Para obter detalhes, confira [Configurar a conectividade TLS no Banco de Dados do Azure para PostgreSQL – Servidor Único](#).

O aplicativo de exemplo neste artigo usa TLS com a `postgres::openssl crate`. A função `postgres::Client::connect` é usada para iniciar a conexão e o programa é encerrado no caso de falha.

```
fn main() {
    let pg_host = std::env::var("POSTGRES_HOST").expect("missing environment variable POSTGRES_HOST");
    let pg_user = std::env::var("POSTGRES_USER").expect("missing environment variable POSTGRES_USER");
    let pg_password = std::env::var("POSTGRES_PASSWORD").expect("missing environment variable POSTGRES_PASSWORD");
    let pg_dbname = std::env::var("POSTGRES_DBNAME").unwrap_or("postgres".to_string());

    let builder = SslConnector::builder(SslMethod::tls()).unwrap();
    let tls_connector = MakeTlsConnector::new(builder.build());

    let url = format!(
        "host={} port=5432 user={} password={} dbname={} sslmode=require",
        pg_host, pg_user, pg_password, pg_dbname
    );
    let mut pg_client = postgres::Client::connect(&url, tls_connector).expect("failed to connect to postgres");
    ...
}
```

Remover e criar tabela

O aplicativo de exemplo usa uma tabela `inventory` simples para demonstrar as operações CRUD (criar, ler, atualizar e excluir).

```
CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);
```

A função `drop_create_table` inicialmente tenta `DROP` a tabela `inventory` antes de criar outra. Isso facilita o aprendizado/a experimentação, pois você sempre começa com um estado conhecido (limpo). O método `Execute` é usado para operações de criação e remoção.

```
const CREATE_QUERY: &str =
    "CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);"

const DROP_TABLE: &str = "DROP TABLE inventory;

fn drop_create_table(pg_client: &mut postgres::Client) {
    let res = pg_client.execute(DROP_TABLE, &[]);
    match res {
        Ok(_) => println!("dropped table"),
        Err(e) => println!("failed to drop table {}", e),
    }
    pg_client
        .execute(CREATE_QUERY, &[])
        .expect("failed to create 'inventory' table");
}
```

Inserir dados

`insert_data` adiciona entradas à tabela `inventory`. Ele cria uma [instrução preparada](#) com a função `Prepare`.

```
const INSERT_QUERY: &str = "INSERT INTO inventory (name, quantity) VALUES ($1, $2) RETURNING id;";

fn insert_data(pg_client: &mut postgres::Client) {

    let prep_stmt = pg_client
        .prepare(&INSERT_QUERY)
        .expect("failed to create prepared statement");

    let row = pg_client
        .query_one(&prep_stmt, &[&"item-1", &42])
        .expect("insert failed");

    let id: i32 = row.get(0);
    println!("inserted item with id {}", id);
    ...
}
```

Observe também o uso do método `prepare_typed`, que permite que os tipos de parâmetros de consulta sejam especificados explicitamente.

```
...
let typed_prep_stmt = pg_client
    .prepare_typed(&INSERT_QUERY, &[Type::VARCHAR, Type::INT4])
    .expect("failed to create prepared statement");

let row = pg_client
    .query_one(&typed_prep_stmt, &[&"item-2", &43])
    .expect("insert failed");

let id: i32 = row.get(0);
println!("inserted item with id {}", id);
...
```

Por fim, um loop `for` é usado para adicionar `item-3`, `item-4` e `item-5` com uma quantidade gerada aleatoriamente para cada.

```
...
for n in 3..=5 {
    let row = pg_client
        .query_one(
            &typed_prep_stmt,
            &[
                &("item-".to_owned() + &n.to_string()),
                &rand::thread_rng().gen_range(10..=50),
            ],
        )
        .expect("insert failed");

    let id: i32 = row.get(0);
    println!("inserted item with id {} ", id);
}
...
```

Consultar dados

A função `query_data` demonstra como recuperar dados da tabela `inventory`. O método `query_one` é usado para obter um item por seu `id`.

```

const SELECT_ALL_QUERY: &str = "SELECT * FROM inventory;";
const SELECT_BY_ID: &str = "SELECT name, quantity FROM inventory where id=$1;";

fn query_data(pg_client: &mut postgres::Client) {

    let prep_stmt = pg_client
        .prepare_typed(&SELECT_BY_ID, &[Type::INT4])
        .expect("failed to create prepared statement");

    let item_id = 1;

    let c = pg_client
        .query_one(&prep_stmt, &[&item_id])
        .expect("failed to query item");

    let name: String = c.get(0);
    let quantity: i32 = c.get(1);
    println!("quantity for item {} = {}", name, quantity);
    ...
}

```

Todas as linhas da tabela de estoque são buscadas usando uma consulta `select * from` com o método de [consulta](#). As linhas retornadas são iteradas para extrair o valor de cada coluna usando o método [Get](#).

TIP

Observe como `get` torna possível especificar a coluna por seu índice numérico na linha ou por seu nome de coluna.

```

...
let items = pg_client
    .query(SELECT_ALL_QUERY, &[])
    .expect("select all failed");

println!("listing items...");

for item in items {
    let id: i32 = item.get("id");
    let name: String = item.get("name");
    let quantity: i32 = item.get("quantity");
    println!(
        "item info: id = {}, name = {}, quantity = {} ",
        id, name, quantity
    );
}
...

```

Atualizar dados

A função `update_date` atualiza aleatoriamente a quantidade de todos os itens. Como a função `insert_data` adicionou 5 linhas, o mesmo é levado em conta no loop `for` – `for n in 1..=5`

TIP

Observe que usamos `query` em vez de `execute` porque pretendemos recuperar o `id` e o recém-gerado `quantity` (usando a cláusula de [RETORNO](#)).

```

const UPDATE_QUERY: &str = "UPDATE inventory SET quantity = $1 WHERE name = $2 RETURNING quantity;";

fn update_data(pg_client: &mut postgres::Client) {
    let stmt = pg_client
        .prepare_typed(&UPDATE_QUERY, &[Type::INT4, Type::VARCHAR])
        .expect("failed to create prepared statement");

    for id in 1..=5 {
        let row = pg_client
            .query_one(
                &stmt,
                &[
                    &rand::thread_rng().gen_range(10..=50),
                    &("item-".to_owned() + &id.to_string()),
                ],
            )
            .expect("update failed");

        let quantity: i32 = row.get("quantity");
        println!("updated item id {} to quantity = {}", id, quantity);
    }
}

```

Excluir dados

Por fim, a função `delete` demonstra como remover um item da tabela `inventory` pelo `id`. O `id` é escolhido aleatoriamente – é um inteiro aleatório entre `1` e `5` (incluindo `5`), pois a função `insert_data` tinha adicionado `5` linhas para começar.

TIP

Observe que usamos `query` em vez de `execute`, pois pretendemos recuperar as informações sobre o item que acabamos de excluir (usando a [cláusula de RETORNO](#)).

```

const DELETE_QUERY: &str = "DELETE FROM inventory WHERE id = $1 RETURNING id, name, quantity;";

fn delete(pg_client: &mut postgres::Client) {
    let stmt = pg_client
        .prepare_typed(&DELETE_QUERY, &[Type::INT4])
        .expect("failed to create prepared statement");

    let item = pg_client
        .query_one(&stmt, &[&rand::thread_rng().gen_range(1..=5)])
        .expect("delete failed");

    let id: i32 = item.get(0);
    let name: String = item.get(1);
    let quantity: i32 = item.get(2);
    println!(
        "deleted item info: id = {}, name = {}, quantity = {}",
        id, name, quantity
    );
}

```

Executar o aplicativo

- Para começar, execute o seguinte comando para clonar o repositório de exemplo:

```
git clone https://github.com/Azure-Samples/azure-postgresql-rust-quickstart.git
```

2. Defina as variáveis de ambiente necessárias com os valores que você copiou do portal do Azure:

```
export POSTGRES_HOST=<server name e.g. my-server.postgres.database.azure.com>
export POSTGRES_USER=<admin username e.g. my-admin-user@my-server>
export POSTGRES_PASSWORD=<admin password>
export POSTGRES_DBNAME=<database name. it is optional and defaults to postgres>
```

3. Para executar o aplicativo, altere para o diretório em que você o clonou e execute `cargo run`:

```
cd azure-postgresql-rust-quickstart
cargo run
```

Você deverá ver uma saída semelhante a esta:

```
dropped 'inventory' table
inserted item with id 1
inserted item with id 2
inserted item with id 3
inserted item with id 4
inserted item with id 5
quantity for item item-1 = 42
listing items...
item info: id = 1, name = item-1, quantity = 42
item info: id = 2, name = item-2, quantity = 43
item info: id = 3, name = item-3, quantity = 11
item info: id = 4, name = item-4, quantity = 32
item info: id = 5, name = item-5, quantity = 24
updated item id 1 to quantity = 27
updated item id 2 to quantity = 14
updated item id 3 to quantity = 31
updated item id 4 to quantity = 16
updated item id 5 to quantity = 10
deleted item info: id = 4, name = item-4, quantity = 16
```

4. Para confirmar, você também pode se conectar ao Banco de Dados do Azure para PostgreSQL usando `psql` e executar consultas no banco de dados, por exemplo:

```
select * from inventory;
```

[Está com problemas? Fale conosco](#)

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Próximas etapas

[Gerenciar o servidor de Banco de Dados do Azure para PostgreSQL usando o portal](#)

[Gerenciar o servidor de Banco de Dados do Azure para PostgreSQL usando a CLI](#)

Não foi possível encontrar o que estava procurando? Fale conosco.

Tutorial: Criar um Banco de Dados do Azure para PostgreSQL – Servidor único usando o portal do Azure

21/05/2021 • 9 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço gerenciado que permite executar, gerenciar e dimensionar os bancos de dados altamente disponíveis do PostgreSQL na nuvem. Usando o Portal do Azure, você pode gerenciar facilmente seu servidor e projetar um banco de dados.

Neste tutorial, você usará o Portal do Azure para aprender a:

- Criar um Banco de Dados do Azure para o servidor PostgreSQL
- Configurar o firewall do servidor
- Use o utilitário [psql](#) para criar um banco de dados
- Carregar dados de exemplo
- Consultar dados
- Atualizar dados
- Restaurar dados

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Criar um Banco de Dados do Azure para o PostgreSQL

Um Banco de Dados do Azure para PostgreSQL é criado com um conjunto definido de [recursos de computação e armazenamento](#). O servidor é criado dentro de um [Grupo de recursos do Azure](#).

Siga estas etapas para criar um Banco de Dados do Azure para o servidor PostgreSQL:

1. Clique em **Criar um recurso** no canto superior esquerdo do portal do Azure.
2. Selecione **Bancos de Dados** na página **Novo** e selecione **Banco de Dados do Azure para PostgreSQL** na página **Bancos de Dados**.

The screenshot shows the Microsoft Azure Marketplace interface. On the left, there's a sidebar with various service categories like Painel, Todos os recursos, and Bancos de dados SQL. A red box highlights the 'Criar um recurso' button at the top. The main area is titled 'Novo' and shows a search bar 'Pesquisar no Marketplace'. Below it, there are tabs for 'Azure Marketplace', 'Ver tudo', and 'Em destaque'. The 'Em destaque' tab is selected. A red box highlights the 'Bancos de dados' category. Underneath, there are several service cards: Banco de Dados SQL, SQL Data Warehouse, Pool de banco de dados elástico do SQL, Banco de Dados do Azure para MySQL, Banco de Dados do Azure para PostgreSQL (versão prévia) (which is also highlighted with a red box), SQL Server 2017 Enterprise Windows Server 2016, Azure Cosmos DB, Banco de dados como serviço para Mobile, Cache Redis, and Data Factory.

3. Selecione a opção de implantação Servidor único.

The screenshot shows the 'Select Azure Database for PostgreSQL deployment option' page. At the top, there's a breadcrumb navigation: Home > New > Select Azure Database for PostgreSQL deployment option. The main title is 'Select Azure Database for PostgreSQL deployment option' with a note 'Microsoft - preview'. Below the title, there's a section titled 'How do you plan to use the service?'. It contains two options: 'Single server' and 'Hyperscale (Citus) server group - PREVIEW'. The 'Single server' option is highlighted with a red box. It includes a description: 'Best for broad range of traditional transactional workloads. Enterprise ready, fully managed community PostgreSQL server with up to 64 vCores, optional geospatial support, full-text search and more.' It has 'Create' and 'Learn more' buttons. The 'Hyperscale (Citus) server group - PREVIEW' option includes a description: 'Best for ultra-high performance and data needs beyond 100GB. Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response. Supports both transactional/operational workloads as well as hybrid transactional analytics workloads.' It also has 'Create' and 'Learn more' buttons. The left sidebar of the Azure portal is visible on the left.

4. Preencha o formulário Básico com as seguintes informações:

Microsoft Azure

Home > New > Select Azure Database for PostgreSQL deployment option > Single server

Single server

Microsoft - PREVIEW

Basics Tags Review + create

Create an Azure Database for PostgreSQL server. [Learn more](#)

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription Select existing... Create new

* Resource group Select existing... Create new

SERVER DETAILS

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

* Server name Enter server name

* Data source None Backup

* Admin username Enter server admin login name

* Password Enter password

* Confirm password Confirm the above password

* Location Australia Central

* Version 10

Compute + storage General purpose
4 vCores, 100 GB storage
Configure server

[Review + create](#) [Next : Tags >>](#)

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Subscription	O nome da sua assinatura	A assinatura do Azure que você deseja usar para o servidor. Se você tiver várias assinaturas, escolha a assinatura para a qual você recebe a cobrança do recurso.
Resource group	<i>myresourcegroup</i>	Um novo nome do grupo de recursos ou um existente de sua assinatura.
Nome do servidor	<i>mydemoserver</i>	Um nome exclusivo que identifica o Banco de Dados do Azure para o servidor PostgreSQL. O nome de domínio <i>postgres.database.azure.com</i> é acrescentado ao nome do servidor fornecido. O servidor pode conter apenas letras minúsculas, números e o caractere de hífen (-). Ele deve conter, pelo menos, 3 até 63 caracteres.
Fonte de dados	<i>Nenhuma</i>	Selecione <i>Nenhum</i> para criar um novo servidor do zero. (Você selecionaria <i>Backup</i> se estivesse criando um servidor de um backup de replicação geográfica de um Banco de Dados do Azure para servidor PostgreSQL existente).

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Nome de usuário do administrador	<i>myadmin</i>	Sua própria conta de logon para uso ao se conectar ao servidor. O nome de logon do administrador não pode ser azure_superuser , azure_pg_admin , admin , administrator , root , guest ou public . Ele não pode começar com pg_ .
Senha	Sua senha	Uma nova senha para a conta do administrador do servidor. Ele deve conter entre 8 e 128 caracteres. A senha precisa conter caracteres de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números (0 a 9) e caracteres não alfanuméricos (!, \$, #, % etc.).
Location	A região mais próxima dos usuários	A localização mais próxima dos usuários.
Versão	A última versão principal	A última versão principal do PostgreSQL, a menos que você tenha requisitos específicos.
Computação + armazenamento	Uso Geral, Gen 5, 2 vCores, 5 GB, 7 dias, Com redundância geográfica	As configurações de computação, armazenamento e backup para o novo servidor. Selecione Configurar servidor . Em seguida, selecione a guia Uso Geral . <i>Gen 5, 4 vCores, 100 GB e 7 dias</i> são os valores padrão de Geração da Computação , vCore , Armazenamento e Período de Retenção de Backup . Você pode deixar esses controles deslizantes como estão ou ajustá-los. Para habilitar os backups do servidor em armazenamento com redundância geográfica, selecione Redundância Geográfica das Opções de Redundância de Backup . Para salvar a seleção desse tipo de preço, selecione OK . A captura de tela a seguir demonstra essas seleções.

NOTE

Considere usar o tipo de preço Básico se computação leve e E/S forem adequadas para sua carga de trabalho. Observe que servidores criados no tipo de preço Básico não podem ser dimensionados mais tarde para Uso Geral ou Otimizado para Memória. Veja a [página de preço](#) para obter mais informações.

Tipo de preço

Básico Até 2 núcleos virtuais com desempenho de E/S variável (1 - 2 núcleos virtuais)	Uso geral Até 64 núcleos virtuais com desempenho de E/S previsível (2 - 64 núcleos virtuais)	Otimizado para memória Até 32 núcleos virtuais de memória otimizada com desempenho de E/S previsível (2 - 32 núcleos virtuais)
---	--	--

Observação: Observe que a alteração no tipo de preço Básico e para ele ou a alteração nas opções de redundância de backup após a criação do servidor não têm suporte.

Geração de computação - Saiba mais sobre a geração de computação [\[?\]](#)

Gen 4 **Gen 5**

núcleo virtual - O que é um vCore? [\[?\]](#)

 4 núcleos virtuais

Armazenamento (tipo: Armazenamento de Uso Geral)

 100 GB

PODE USAR ATÉ **300 IOPS** disponíveis

Período de Retenção de Backup

 7 dias

Opções de redundância de backup - Saiba mais detalhes [\[?\]](#)

Com redundância local
Recuperar da perda de dados na região

Georredundante
Recuperar da interrupção regional ou desastre

OK

RESUMO DE PREÇO

Geração da computação Gen5

Custo por **núcleo virtual** **76,19**

núcleos virtuais selecionados **x 4**

Armazenamento de uso geral

Custo por **GB / mês** **0,14**

Quantidade de armazenamento selecionada **x 100**

EST. CUSTO MENSAL **318,54 USD**

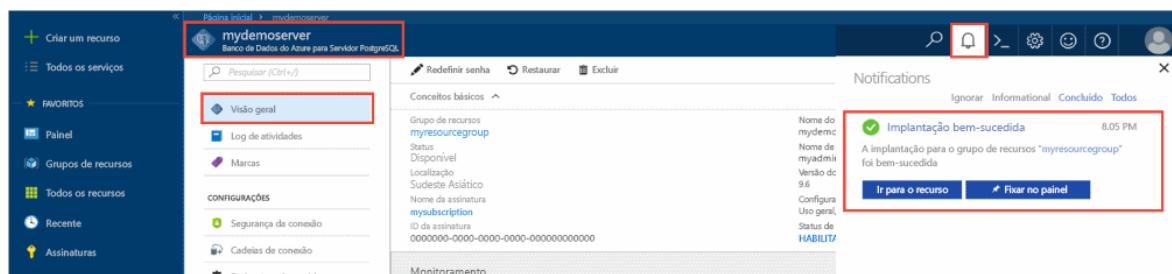
COBRANÇA ADICIONAL POR USO

Consulte os [detalhes de preço](#) para saber mais detalhes.

TIP

Com o **aumento automático** habilitado, seu servidor aumenta o armazenamento quando você está se aproximando do limite alocado, sem afetar sua carga de trabalho.

5. Selecione **Revisar + criar** para revisar suas seleções. Selecione **Criar** para provisionar o servidor. Esta operação pode levar alguns minutos.
6. Na barra de ferramentas, selecione o ícone (sino) **Notificações** para monitorar o processo de implantação. Depois que a implantação é feita, você pode selecionar **Fixar no painel**, que cria um bloco para esse servidor no seu painel do portal do Azure como um atalho para a página **Visão geral** do servidor. A opção **Ir para recurso** abre a página **Visão geral** do servidor.



The screenshot shows the Azure portal interface for provisioning a PostgreSQL server. The left sidebar includes 'Criar um recurso', 'Todos os serviços', 'FAVORITOS' (with 'Painel' selected), 'Grupos de recursos', 'Todos os recursos', 'Recente', and 'Assinaturas'. The main area shows the 'mydemoserver' resource group with a 'Visão geral' card. The 'Notifications' pane displays a success message: 'Implantação bem-sucedida' at 8:05 PM, with the link 'Ir para o recurso' also highlighted with a red box.

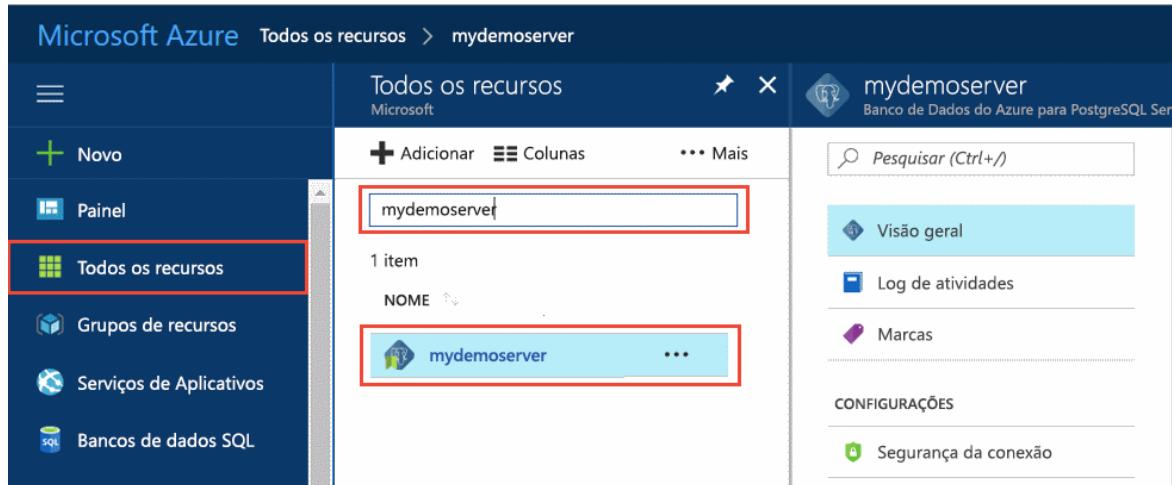
Por padrão, um banco de dados **postgres** é criado no servidor. O banco de dados **postgres** é um banco de dados padrão destinado ao uso dos usuários, de utilitários e de aplicativos de terceiros. (O outro banco de dados padrão é o **azure_maintenance**. Sua função é separar os processos de serviço gerenciado das ações do usuário. Não é possível acessar este banco de dados).

Configurar uma regra de firewall no nível de servidor

O serviço do Banco de Dados do Azure para PostgreSQL cria um firewall no nível do servidor. Por padrão, esse

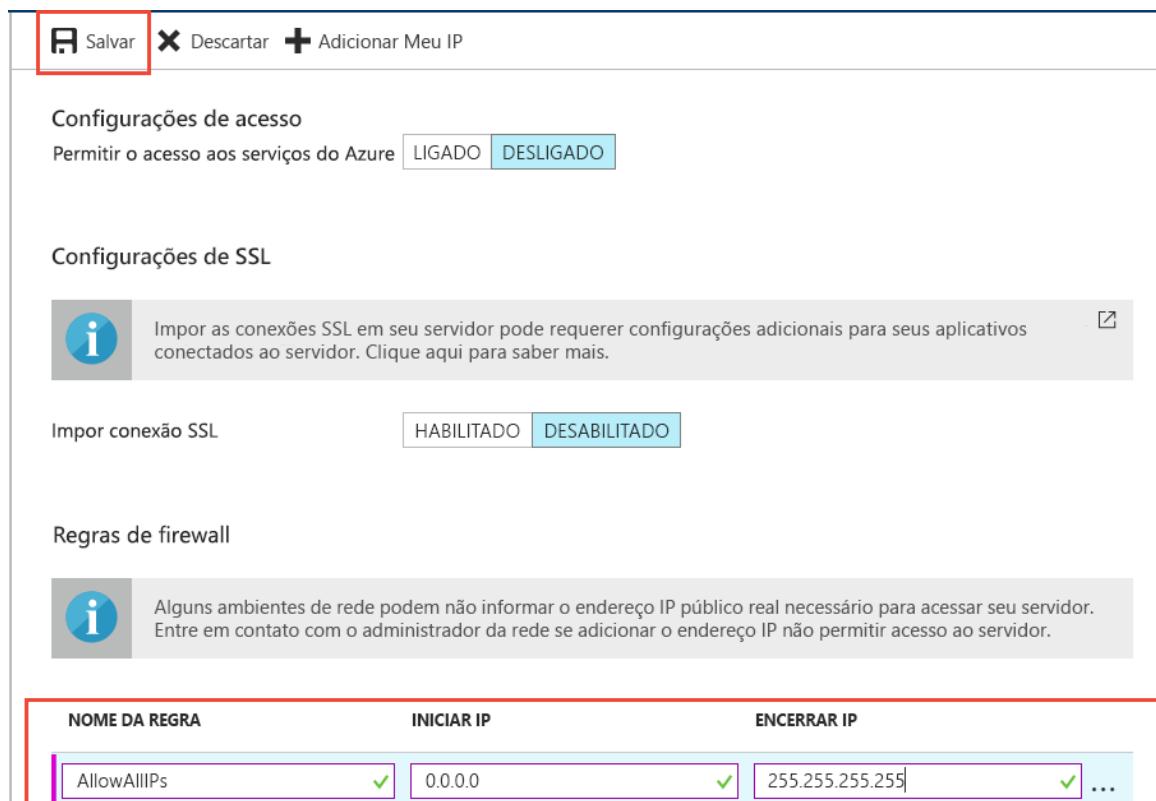
firewall impede que todos os aplicativos e ferramentas externos se conectem ao servidor e a todos os bancos de dados no servidor, a menos que uma regra de firewall seja criada para abrir o firewall para um intervalo de endereços IP específico.

1. Após a implantação ser concluída, clique em **Todos os Recursos** no menu esquerdo e digite o nome **mydemoserver**, para pesquisar o servidor recém-criado. Clique no nome do servidor listado nos resultados da pesquisa. A página **Visão geral** do servidor é aberta e oferece outras opções de configuração.



The screenshot shows the Microsoft Azure portal interface. On the left sidebar, 'Todos os recursos' is selected and highlighted with a red box. In the main search results area, the search term 'mydemoserve' is shown in the search bar, with the last character 'r' redacted. Below it, a single item named 'mydemoserver' is listed, also highlighted with a red box. To the right, the 'Visão geral' (General View) of the server is displayed, along with other options like 'Log de atividades' (Activity Log) and 'Marcas' (Tags).

2. Na página do servidor, selecione **Segurança da conexão**.
3. Clique na caixa de texto em **Nome da regra** e adicione uma nova regra de firewall para especificar o intervalo de IP para conectividade. Insira o intervalo de IP. Clique em **Save (Salvar)**.



The screenshot shows the 'Segurança da conexão' (Connection Security) page for the 'mydemoserver' database. At the top, there are buttons for 'Salvar' (Save), 'Descartar' (Discard), and 'Adicionar Meu IP'. Below, under 'Configurações de acesso' (Access Settings), there is a note about allowing Azure services access, with a switch set to 'LIGADO' (ON). Under 'Configurações de SSL' (SSL Settings), there is a note about importing SSL connections, with a switch set to 'DESABILITADO' (OFF). In the 'Regras de firewall' (Firewall Rules) section, a table lists a single rule: 'AllowAllIPs' with 'INICIAR IP' set to '0.0.0.0' and 'ENCERRAR IP' set to '255.255.255.255'.

4. Clique em **Salvar** e, em seguida, clique no X para fechar a página de **Segurança de conexões**.

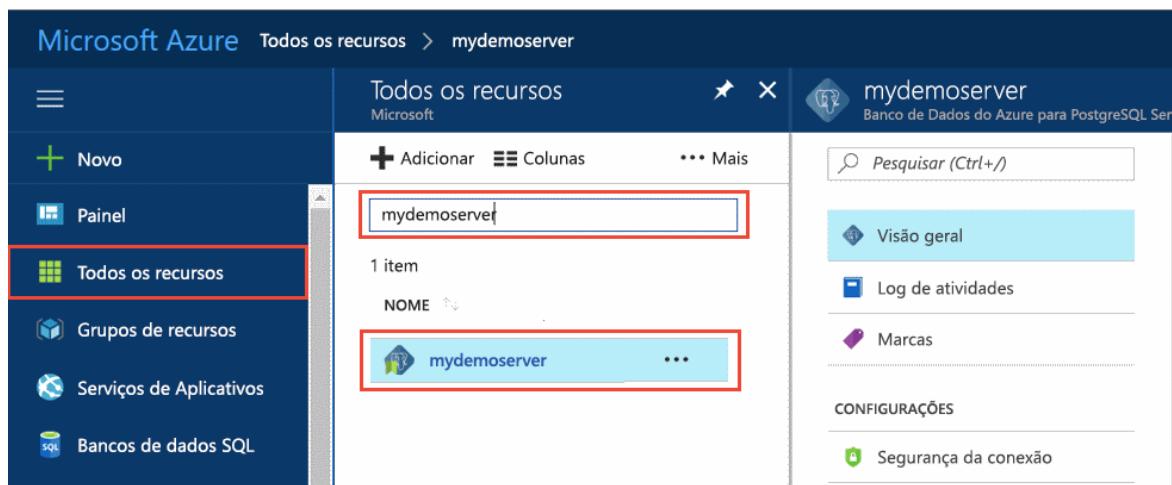
NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Se você estiver tentando se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Se isso acontecer, você não poderá conectar o servidor do Banco de Dados SQL do Azure, a menos que o departamento de TI abra a porta 5432.

Obter informações de conexão

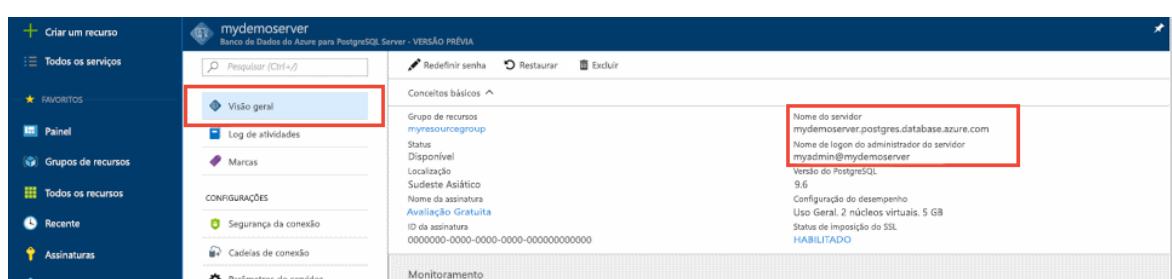
Quando você criou o servidor do Banco de dados do Azure para PostgreSQL, o banco de dados **postgres** padrão também foi criado. Para se conectar ao seu servidor de banco de dados, você precisa fornecer credenciais de acesso e informações de host.

1. No menu à esquerda no portal do Azure, clique em **Todos os recursos** e pesquise pelo servidor que você acabou de criar.



The screenshot shows the Microsoft Azure portal interface. On the left sidebar, the 'Todos os recursos' option is selected and highlighted with a red box. In the center, a search bar also has 'mydemoserver' typed into it and is highlighted with a red box. Below the search bar, the results show one item: 'mydemoserver' with its icon and name highlighted with a red box. To the right, there's a detailed view of the server 'mydemoserver' under the 'Banco de Dados do Azure para PostgreSQL Server' category. The 'Visão geral' tab is selected and highlighted with a blue box. Other tabs include 'Log de atividades' and 'Marcas'. Under 'CONFIGURAÇÕES', the 'Segurança da conexão' tab is visible.

2. Clique no nome do servidor **mydemoserver**.
3. Selecione a página **Visão geral** do servidor. Anote o **Nome do servidor** e o **Nome de logon de administrador do servidor**.



The screenshot shows the 'Visão geral' (General View) page for the 'mydemoserver' PostgreSQL server. The 'Visão geral' tab is selected and highlighted with a red box. In the main content area, there are two fields highlighted with red boxes: 'Nome do servidor' (mydemoserver.postgres.database.azure.com) and 'Nome de logon do administrador do servidor' (myadmin@mydemoserver). Other details shown include the group resource ('myresourcegroup'), status ('Disponível'), location ('Sudeste Asiático'), and PostgreSQL version ('9.6').

Conectar-se ao banco de dados PostgreSQL usando psql

Se o computador cliente tiver o PostgreSQL instalado, você poderá usar uma instância local do [psql](#), ou o Azure Cloud Console para se conectar a um servidor PostgreSQL do Azure. Usaremos agora o utilitário de linha de comando `psql` para nos conectarmos ao Banco de Dados do Azure para o servidor PostgreSQL.

1. Execute o comando `psql` a seguir para se conectar a um Banco de Dados do Azure para PostgreSQL:

```
psql --host=<servername> --port=<port> --username=<user@servername> --dbname=<dbname>
```

Por exemplo, o comando a seguir se conecta ao banco de dados padrão chamado **postgres** no seu servidor PostgreSQL **mydemoserver.postgres.database.azure.com** usando as credenciais de acesso.

Insira o <server_admin_password> que você escolheu quando uma senha foi solicitada a você.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --dbname=postgres
```

TIP

Se você preferir usar um caminho de URL para se conectar ao Postgres, codifique a URL de entrada @ do nome de usuário com %40. Por exemplo, a cadeia de conexão para psql seria,

```
psql postgresql://myadmin%40mydemoserver@mydemoserver.postgres.database.azure.com:5432/postgres
```

2. Quando já estiver conectado ao servidor, crie um banco de dados em branco no prompt:

```
CREATE DATABASE mypgsqlldb;
```

3. No prompt, execute o seguinte comando para mudar a conexão para o banco de dados **mypgsqlldb** recém-criado:

```
\c mypgsqlldb
```

Criar tabelas no banco de dados

Agora que você sabe como se conectar ao Banco de Dados do Azure para PostgreSQL, pode concluir algumas tarefas básicas:

Primeiro, crie uma tabela e carregue-a com alguns dados. Vamos criar uma tabela que acompanha as informações de inventário usando este código SQL:

```
CREATE TABLE inventory (
    id serial PRIMARY KEY,
    name VARCHAR(50),
    quantity INTEGER
);
```

Você agora pode ver a tabela recém-criada na lista de tabelas digitando:

```
\dt
```

Carregar dados nas tabelas

Agora que você tem uma tabela, insira alguns dados nela. Na janela do prompt de comando aberta, execute a consulta a seguir para inserir algumas linhas de dados.

```
INSERT INTO inventory (id, name, quantity) VALUES (1, 'banana', 150);
INSERT INTO inventory (id, name, quantity) VALUES (2, 'orange', 154);
```

Você tem agora duas linhas de dados de exemplo na tabela de inventário criada anteriormente.

Consultar e atualizar os dados nas tabelas

Execute a seguinte consulta para recuperar as informações da tabela do banco de dados de inventário.

```
SELECT * FROM inventory;
```

Também é possível atualizar os dados na tabela.

```
UPDATE inventory SET quantity = 200 WHERE name = 'banana';
```

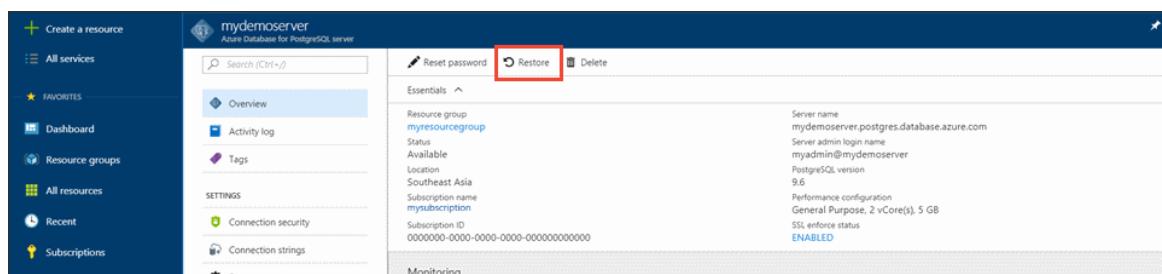
Você pode ver os valores atualizados quando recuperar os dados.

```
SELECT * FROM inventory;
```

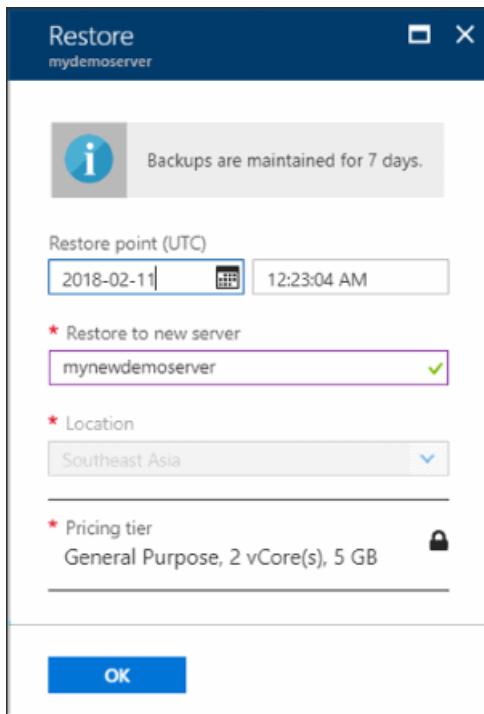
Restaurar dados para um ponto anterior no tempo

Imagine que você excluiu acidentalmente essa tabela. Essa situação é algo do qual você não pode se recuperar facilmente. O Banco de Dados do Azure para PostgreSQL permite que você volte para qualquer ponto anterior no qual o servidor tenha backups (determinados pelo período de retenção de backup que você configurou) e restaure esse ponto como um novo servidor. Use esse novo servidor para recuperar seus dados excluídos. As etapas a seguir restauram o servidor **mydemoserver** para um ponto anterior à adição da tabela de inventário.

1. Na página de **Visão geral** do Banco de Dados do Azure para PostgreSQL para o servidor, clique em **Restaurar** na barra de ferramentas. A página **Restaurar** será aberta.



2. Preencha o formulário **Restaurar** com as informações necessárias:



- **Ponto de restauração:** Selecione um ponto no tempo anterior à alteração do servidor
- **Servidor de destino:** Forneça o novo nome do servidor para o qual deseja fazer a restauração
- **Localização:** Não é possível selecionar a região; por padrão, ela é a mesma do servidor de origem
- **Tipo de preço:** Não é possível alterar esse valor ao restaurar um servidor. Ele é igual ao servidor de origem.

3. Clique em **OK** para [restaurar o servidor para um ponto](#) anterior à exclusão da tabela. A restauração de um servidor para um ponto diferente no tempo cria um novo servidor duplicado como o servidor original a partir do ponto no tempo especificado por você, desde que esteja dentro do período de retenção do seu [tipo de preço](#).

Limpar os recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão *Excluir* na página *Visão geral* do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão *Excluir* final.

Próximas etapas

Neste tutorial, você aprendeu a usar o portal do Azure e outros utilitários para:

- Criar um Banco de Dados do Azure para o servidor PostgreSQL
- Configurar o firewall do servidor
- Use o utilitário `psql` para criar um banco de dados
- Carregar dados de exemplo
- Consultar dados
- Atualizar dados
- Restaurar dados

[Criar seu primeiro Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#)

Tutorial: Criar um Banco de Dados do Azure para PostgreSQL – Servidor único usando a CLI do Azure

21/05/2021 • 9 minutes to read

Neste tutorial, você usará a CLI (interface de linha de comando) do Azure e outros utilitários para aprender a:

- Criar um Banco de Dados do Azure para o servidor PostgreSQL
- Configurar o firewall do servidor
- Use o utilitário [psql](#) para criar um banco de dados
- Carregar dados de exemplo
- Consultar dados
- Atualizar dados
- Restaurar dados

Você pode usar o Azure Cloud Shell no navegador ou [instalar a CLI do Azure](#) em seu computador para executar os comandos neste tutorial.

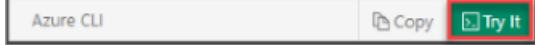
Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou

Cmd+Shift+V no macOS.

4. Pressione **Enter** para executar o código.

Se você optar por instalar e usar a CLI localmente, este artigo exigirá que seja executada a CLI do Azure versão 2.0 ou posterior. Execute `az --version` para encontrar a versão. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Se tiver várias assinaturas, escolha a que for adequada na qual existe o recurso ou onde ele é cobrado. Selecione uma ID da assinatura específica em sua conta usando o comando `az account set`.

```
az account set --subscription 00000000-0000-0000-0000-000000000000
```

Criar um grupo de recursos

Crie um [grupo de recursos do Azure](#) usando o comando `az group create`. Um grupo de recursos é um contêiner lógico no qual os recursos do Azure são implantados e gerenciados em grupo. O exemplo a seguir cria um grupo de recursos chamado `myresourcegroup` na localização `westus`.

```
az group create --name myresourcegroup --location westus
```

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Crie um [Banco de Dados do Azure para PostgreSQL](#) usando o comando `az postgres server create`. Um servidor contém um grupo de bancos de dados gerenciados conjuntamente.

O exemplo a seguir cria um servidor chamado `mydemoserver` em seu grupo de recursos `myresourcegroup` com o logon de administrador de servidor `myadmin`. O nome de um servidor é mapeado para o nome DNS e, portanto, deve ser globalmente exclusivo no Azure. Substitua o `<server_admin_password>` com seu próprio valor. É um servidor de Geração 5 de Uso Geral com 2 vCores.

```
az postgres server create --resource-group myresourcegroup --name mydemoserver --location westus --admin-user myadmin --admin-password <server_admin_password> --sku-name GP_Gen5_2 --version 9.6
```

O valor do parâmetro `sku-name` segue a convenção {camada de preços}_{geração de cálculo}_{vCores} como nestes exemplos:

- `--sku-name B_Gen5_2` mapeia para Básico, Gen 5 e 2 vCores.
- `--sku-name GP_Gen5_32` mapeia para Uso Geral, Gen 5 e 32 vCores.
- `--sku-name MO_Gen5_2` mapeia para Otimizado para Memória, Gen 5 e 2 vCores.

Veja a documentação das [camadas de preços](#) para entender os valores válidos por região e por camada.

IMPORTANT

O logon de administrador do servidor e a senha que você especificar aqui são necessários para fazer logon no servidor e em seus bancos de dados mais tarde neste Guia de início rápido. Lembre-se ou registre essas informações para o uso posterior.

Por padrão, o banco de dados `postgres` é criado em seu servidor. O `postgres` é um banco de dados padrão destinado a uso por usuários, utilitários e aplicativos de terceiros.

Configurar uma regra de firewall no nível de servidor

Crie uma regra de firewall no nível de servidor do PostgreSQL do Azure com o comando [az postgres server firewall-rule create](#). Uma regra de firewall de nível de servidor permite que um aplicativo externo, tal como `psql` ou [PgAdmin](#), conecte-se ao servidor por meio do firewall do serviço Azure PostgreSQL.

Você pode definir uma regra de firewall que abranja um intervalo de IP aos quais você possa se conectar de sua rede. O exemplo a seguir usa [az postgres server firewall-rule create](#) para criar uma regra de firewall `AllowMyIP` que permite a conexão de um único endereço IP.

```
az postgres server firewall-rule create --resource-group myresourcegroup --server mydemoserver --name AllowMyIP --start-ip-address 192.168.0.1 --end-ip-address 192.168.0.1
```

Para restringir o acesso ao seu servidor do Azure PostgreSQL para sua rede apenas, você pode definir a regra de firewall para abranger apenas o intervalo de endereços IP de sua rede corporativa.

NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Ao se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Faça seu departamento de TI abrir a porta 5432 para se conectar ao seu servidor do Banco de Dados SQL do Azure.

Obter informações de conexão

Para se conectar ao servidor, é preciso fornecer credenciais de acesso e informações do host.

```
az postgres server show --resource-group myresourcegroup --name mydemoserver
```

O resultado está no formato JSON. Anote o `administratorLogin` e o `fullyQualifiedDomainName`.

```
{
  "administratorLogin": "myadmin",
  "earliestRestoreDate": null,
  "fullyQualifiedDomainName": "mydemoserver.postgres.database.azure.com",
  "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/myresourcegroup/providers/Microsoft.DBforPostgreSQL/servers/mydemoserver",
  "location": "westus",
  "name": "mydemoserver",
  "resourceGroup": "myresourcegroup",
  "sku": {
    "capacity": 2,
    "family": "Gen5",
    "name": "GP_Gen5_2",
    "size": null,
    "tier": "GeneralPurpose"
  },
  "sslEnforcement": "Enabled",
  "storageProfile": {
    "backupRetentionDays": 7,
    "geoRedundantBackup": "Disabled",
    "storageMb": 5120
  },
  "tags": null,
  "type": "Microsoft.DBforPostgreSQL/servers",
  "userVisibleState": "Ready",
  "version": "9.6"
}
```

Conectar-se ao Banco de Dados do Azure para o banco de dados PostgreSQL usando psql

Se o computador cliente tiver o PostgreSQL instalado, você poderá usar uma instância local do `psql`, ou o Azure Cloud Console para se conectar a um servidor PostgreSQL do Azure. Usaremos agora o utilitário de linha de comando `psql` para nos conectarmos ao Banco de Dados do Azure para o servidor PostgreSQL.

- Execute o comando `psql` a seguir para se conectar a um Banco de Dados do Azure para PostgreSQL:

```
psql --host=<servername> --port=<port> --username=<user@servername> --dbname=<dbname>
```

Por exemplo, o comando a seguir se conecta ao banco de dados padrão chamado `postgres` no seu servidor PostgreSQL `mydemoserver.postgres.database.azure.com` usando as credenciais de acesso.

Insira o `<server_admin_password>` que você escolheu quando uma senha foi solicitada a você.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --
dbname=postgres
```

TIP

Se você preferir usar um caminho de URL para se conectar ao Postgres, codifique a URL de entrada @ do nome de usuário com `%40`. Por exemplo, a cadeia de conexão para `psql` seria,

```
psql postgresql://myadmin%40mydemoserver@mydemoserver.postgres.database.azure.com:5432/postgres
```

- Quando já estiver conectado ao servidor, crie um banco de dados em branco no prompt:

```
CREATE DATABASE mypgsqlDb;
```

3. No prompt, execute o seguinte comando para mudar a conexão para o banco de dados **mypgsqlDb** recém-criado:

```
\c mypgsqlDb
```

Criar tabelas no banco de dados

Agora que você sabe como se conectar ao Banco de Dados do Azure para PostgreSQL, pode concluir algumas tarefas básicas:

Primeiro, crie uma tabela e carregue-a com alguns dados. Por exemplo, crie uma tabela que rastreia informações de inventário:

```
CREATE TABLE inventory (
    id serial PRIMARY KEY,
    name VARCHAR(50),
    quantity INTEGER
);
```

Você agora pode ver a tabela recém-criada na lista de tabelas digitando:

```
\dt
```

Carregar dados na tabela

Agora que uma tabela foi criada, insira alguns dados nela. Na janela do prompt de comando aberta, execute a consulta a seguir para inserir algumas linhas de dados:

```
INSERT INTO inventory (id, name, quantity) VALUES (1, 'banana', 150);
INSERT INTO inventory (id, name, quantity) VALUES (2, 'orange', 154);
```

Agora, você adicionou duas linhas de dados de exemplo na tabela criada anteriormente.

Consultar e atualizar os dados nas tabelas

Execute a seguinte consulta para recuperar as informações da tabela do inventário:

```
SELECT * FROM inventory;
```

Também é possível atualizar os dados na tabela do inventário:

```
UPDATE inventory SET quantity = 200 WHERE name = 'banana';
```

Você pode ver os valores atualizados quando recuperar os dados:

```
SELECT * FROM inventory;
```

Restaurar um banco de dados em um ponto anterior no tempo

Imagine que você excluiu acidentalmente uma tabela. Isso é algo que você não pode se recuperar facilmente. O Banco de Dados do Azure para PostgreSQL permite que você volte para qualquer ponto anterior no qual o servidor tenha backups (determinados pelo período de retenção de backup que você configurou) e restaure esse ponto como um novo servidor. Use esse novo servidor para recuperar seus dados excluídos.

O comando a seguir restaura o servidor de exemplo para um ponto anterior à adição da tabela:

```
az postgres server restore --resource-group myresourcegroup --name mydemoserver-restored --restore-point-in-time 2017-04-13T13:59:00Z --source-server mydemoserver
```

O comando `az postgres server restore` precisa dos seguintes parâmetros:

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
resource-group	myresourcegroup	O grupo de recursos no qual o servidor de origem existe.
name	mydemoserver-restored	O nome do novo servidor que é criado pelo comando de restauração.
restore-point-in-time	2017-04-13T13:59:00Z	Selecione um point-in-time para restaurar. Essa data e hora devem estar dentro do período de retenção de backup do servidor de origem. Use o formato ISO8601 de data e hora. Por exemplo, você pode usar seu próprio fuso horário local, como <code>2017-04-13T05:59:00-08:00</code> , ou usar o formato UTC Zulu <code>2017-04-13T13:59:00Z</code> .
source-server	mydemoserver	O nome ou ID para restaurar a partir do servidor de origem.

Restaurar um servidor para um ponto no tempo cria um novo servidor, copiado como o servidor original do ponto no tempo que você especificar. O local e os valores de tipo de preço para o servidor restaurado são o mesmo que o servidor de origem.

O comando é síncrono e retornará depois que o servidor for restaurado. Depois que a restauração é concluída, localize o novo servidor que foi criado. Verifique se os dados foram restaurados conforme o esperado.

Limpar os recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão *Excluir* na página *Visão geral* do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão *Excluir* final.

Próximas etapas

Neste tutorial, você aprendeu como usar a CLI (interface de linha de comando) do Azure e outros utilitários para aprender a:

- Criar um Banco de Dados do Azure para o servidor PostgreSQL

- Configurar o firewall do servidor
- Use o utilitário `psql` para criar um banco de dados
- Carregar dados de exemplo
- Consultar dados
- Atualizar dados
- Restaurar dados

[Criar seu primeiro Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#)

Tutorial: Criar um Banco de Dados do Azure para PostgreSQL – Servidor único usando o PowerShell

21/05/2021 • 8 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço de banco de dados relacional no Microsoft Cloud, baseado no mecanismo de banco de dados PostgreSQL Community Edition. Neste tutorial, você usa o PowerShell e outros utilitários para aprender a:

- Criar um Banco de Dados do Azure para o PostgreSQL
- Configurar o firewall do servidor
- Use o utilitário `psql` para criar um banco de dados
- Carregar dados de exemplo
- Consultar dados
- Atualizar dados
- Restaurar dados

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Se você optar por usar o PowerShell localmente, este artigo exigirá que você instale o módulo Az PowerShell e conecte-se à sua conta do Azure usando o cmdlet `Connect-AzAccount`. Para obter mais informações sobre como instalar o módulo Az PowerShell, confira [Instalar o Azure PowerShell](#).

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Se esta é a primeira vez que você usa o serviço de Banco de Dados do Azure para PostgreSQL, registre o provedor de recursos **Microsoft.DBforPostgreSQL**.

```
Register-AzResourceProvider -ProviderNamespace Microsoft.DBforPostgreSQL
```

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

Se tiver várias assinaturas do Azure, escolha a que for adequada para cobrança do recurso. Selecione uma ID de assinatura específica usando o cmdlet [Set-AzContext](#).

```
Set-AzContext -SubscriptionId 00000000-0000-0000-0000-000000000000
```

Criar um grupo de recursos

Crie um [grupo de recursos do Azure](#) usando o cmdlet [New-AzResourceGroup](#). Um grupo de recursos é um contêiner lógico no qual os recursos do Azure são implantados e gerenciados como um grupo.

O seguinte exemplo cria um grupo de recursos chamado **myresourcegroup** na região **Oeste dos EUA**.

```
New-AzResourceGroup -Name myresourcegroup -Location westus
```

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Crie um Banco de Dados do Azure para PostgreSQL com o cmdlet [New-AzPostgreSQLServer](#). Um servidor pode gerenciar vários bancos de dados. Normalmente, um banco de dados separado é usado para cada projeto ou para cada usuário.

O exemplo a seguir cria um servidor PostgreSQL na região **Oeste dos EUA** denominada **mydemoserver** no grupo de recursos **myresourcegroup** com um logon de administrador do servidor de **myadmin**. É um servidor Gen 5 no tipo de preço de uso geral com dois vCores e backups com redundância geográfica habilitados. Documente a senha usada na primeira linha do exemplo, pois essa é a senha da conta do administrador do servidor PostgreSQL.

TIP

Um nome de servidor mapeia para um nome DNS e deve ser globalmente exclusivo no Azure.

```
$Password = Read-Host -Prompt 'Please enter your password' -AsSecureString
New-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup -Sku GP_Gen5_2 -
GeoRedundantBackup Enabled -Location westus -AdministratorUsername myadmin -AdministratorLoginPassword
$Password
```

O valor do parâmetro **SKU** segue a convenção **pricing-tier_compute-generation_vCores**, conforme mostrado nestes exemplos.

- `-Sku B_Gen5_1` é mapeado para Básico, Gen 5 e 1 vCore. Essa opção é o menor SKU disponível.
- `-Sku GP_Gen5_32` mapeia para Uso Geral, Gen 5 e 32 vCores.
- `-Sku M0_Gen5_2` mapeia para Otimizado para Memória, Gen 5 e 2 vCores.

Para obter informações sobre valores de **SKU** válidos por região e para camadas, confira [Tipos de preço do Banco de Dados do Azure para PostgreSQL](#).

Considere usar o tipo de preço Básico se computação leve e E/S forem adequadas para sua carga de trabalho.

IMPORTANT

Os servidores criados no tipo de preço básico não podem depois ser dimensionados para uso geral ou com otimização de memória e não podem ser replicados geograficamente.

Configurar uma regra de firewall

Crie uma regra de firewall no nível de servidor do Banco de Dados do Azure para PostgreSQL usando o cmdlet `New-AzPostgreSqlServerFirewallRule`. Uma regra de firewall no nível de servidor permite que um aplicativo externo, como a ferramenta de linha de comando `psql` ou o PostgreSQL Workbench, conecte-se ao servidor por meio do firewall do serviço Banco de Dados do Azure para PostgreSQL.

O exemplo a seguir cria uma regra de firewall chamada **AllowMyIP**, que permite conexões de um endereço IP específico, 192.168.0.1. Substitua um endereço IP ou um intervalo de endereços IP que correspondam à localização da qual você está se conectando.

```
New-AzPostgreSqlServerFirewallRule -Name AllowMyIP -ResourceGroupName myresourcegroup -ServerName mydemoserver -
StartIPAddress 192.168.0.1 -EndIPAddress 192.168.0.1
```

NOTE

As conexões ao Banco de Dados do Azure para PostgreSQL se comunicam pela porta 5432. Se estiver tentando se conectar em uma rede corporativa, talvez o tráfego de saída pela porta 5432 não seja permitido. Nesse cenário, você só poderá se conectar ao servidor se o departamento de TI abrir a porta 5432.

Obter informações de conexão

Para se conectar ao servidor, é preciso fornecer credenciais de acesso e informações do host. Use o exemplo a seguir para determinar as informações de conexão. Anote os valores de **FullyQualifiedDomainName** e o **AdministratorLogin**.

```
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |
Select-Object -Property FullyQualifiedDomainName, AdministratorLogin
```

FullyQualifiedDomainName	AdministratorLogin
mydemoserver.postgresql.database.azure.com	myadmin

Conectar-se ao banco de dados PostgreSQL usando psql

Se o computador cliente tiver o PostgreSQL instalado, você poderá usar uma instância local do `psql` para se conectar a um servidor PostgreSQL do Azure. Você também pode acessar uma versão pré-instalada da ferramenta de linha de comando `psql` no Azure Cloud Shell selecionando o botão **Experimentar** em um exemplo de código neste artigo. Outras maneiras de acessar o Azure Cloud Shell são selecionar o botão `>_` na barra de ferramentas superior direita no portal do Azure ou acessar shell.azure.com.

1. Conecte-se ao servidor PostgreSQL do Azure usando o utilitário de linha de comando `psql`.

```
psql --host=<servername> --port=<port> --username=<user@servername> --dbname=<dbname>
```

Por exemplo, o comando a seguir se conecta ao banco de dados padrão chamado `postgres` no seu servidor PostgreSQL `mydemoserver.postgres.database.azure.com` usando as credenciais de acesso. Insira o `<server_admin_password>` que você escolheu quando uma senha foi solicitada a você.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin@mydemoserver --dbname=postgres
```

TIP

Se você preferir usar um caminho de URL para se conectar ao Postgres, codifique a URL de entrada @ do nome de usuário com `%40`. Por exemplo, a cadeia de conexão para psql seria

```
psql postgresql://myadmin%40mydemoserver@mydemoserver.postgres.database.azure.com:5432/postgres
```

2. Quando já estiver conectado ao servidor, crie um banco de dados em branco no prompt.

```
CREATE DATABASE mypgsqlldb;
```

3. No prompt, execute o seguinte comando para mudar a conexão para o banco de dados `mypgsqlldb` recém-criado:

```
\c mypgsqlldb
```

Criar tabelas no banco de dados

Agora que você sabe como se conectar ao Banco de Dados do Azure para PostgreSQL, conclua algumas tarefas básicas.

Primeiro, crie uma tabela e carregue-a com alguns dados. Vamos criar uma tabela que armazena informações de inventário.

```
CREATE TABLE inventory (
    id serial PRIMARY KEY,
    name VARCHAR(50),
    quantity INTEGER
);
```

Carregar dados nas tabelas

Agora que você tem uma tabela, insira alguns dados nela. Na janela do prompt de comando aberta, execute a consulta a seguir para inserir algumas linhas de dados.

```
INSERT INTO inventory (id, name, quantity) VALUES (1, 'banana', 150);
INSERT INTO inventory (id, name, quantity) VALUES (2, 'orange', 154);
```

Agora, você tem duas linhas de dados de exemplo na tabela criada anteriormente.

Consultar e atualizar os dados nas tabelas

Execute a seguinte consulta para recuperar as informações da tabela do banco de dados.

```
SELECT * FROM inventory;
```

Também é possível atualizar os dados nas tabelas.

```
UPDATE inventory SET quantity = 200 WHERE name = 'banana';
```

A linha é atualizada à medida que você recupera os dados.

```
SELECT * FROM inventory;
```

Restaurar um banco de dados em um ponto anterior no tempo

Você pode restaurar o servidor para um ponto anterior no tempo. Os dados restaurados são copiados para um novo servidor e o servidor existente é deixado como está. Por exemplo, se uma tabela for removida acidentalmente, você poderá restaurar até o momento em que a remoção ocorreu. Depois, você pode recuperar a tabela e os dados ausentes da cópia restaurada do servidor.

Para restaurar o servidor, use o cmdlet `Restore-AzPostgreSqlServer` do PowerShell.

Executar o comando restore

Para restaurar o servidor, execute o exemplo do PowerShell a seguir.

```
$restorePointInTime = (Get-Date).AddMinutes(-10)
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |
    Restore-AzPostgreSqlServer -Name mydemoserver-restored -ResourceGroupName myresourcegroup -
        RestorePointInTime $restorePointInTime -UsePointInTimeRestore
```

Quando você restaura um servidor para um ponto anterior no tempo, é criado um servidor. O servidor original e seus bancos de dados do ponto no tempo especificado são copiados para o novo servidor.

Os valores de local e tipo de preço para o servidor restaurado permanecem iguais aos do servidor de origem.

Depois que o processo de restauração é concluído, localize o novo servidor e verifique se os dados são restaurados como esperado. O novo servidor tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no momento em que a restauração foi iniciada. A senha pode ser alterada na página **Visão geral** do servidor.

O servidor criado durante uma restauração não tem o ponto de extremidade de serviço VNet existentes no servidor original. Essas regras devem ser configuradas separadamente para o novo servidor. As regras de firewall do servidor original são restauradas.

Limpar recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão *Excluir* na página *Visão geral* do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão *Excluir* final.

Próximas etapas

[Como fazer backup e restaurar um servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell](#)

Tutorial: Implantar um aplicativo Web Django com o PostgreSQL no Serviço de Aplicativo do Azure

16/06/2021 • 16 minutes to read

Este tutorial mostra como implantar um aplicativo Web Python [Django](#) controlado por dados no [Serviço de Aplicativo do Azure](#) e conectá-lo a um Banco de Dados do Azure para PostgreSQL. O Serviço de Aplicativo fornece um serviço de hospedagem Web altamente escalonável e com aplicação automática de patch.

Neste tutorial, você usa a CLI do Azure para concluir as seguintes tarefas:

- Configurar o ambiente inicial com Python e a CLI do Azure
- Criar um Banco de Dados do Azure para PostgreSQL
- Implantar um código no Serviço de Aplicativo do Azure e conectar-se ao PostgreSQL
- Atualizar seu código e reimplantar
- Exibir logs de diagnóstico
- Gerenciar o aplicativo Web no portal do Azure

Use também a [versão do portal do Azure deste tutorial](#).

1. Configurar o seu ambiente inicial

1. Tenha uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
2. Instale o [Python 3.6 ou mais recente](#).
3. Instale a [CLI do Azure](#) 2.18.0 ou superior, com a qual você executa comandos em qualquer shell para provisionar e configurar recursos do Azure.

Abra uma janela de terminal e verifique se sua versão do Python é 3.6 ou superior:

- [Bash](#)
- [PowerShell](#)
- [Cmd](#)

```
python3 --version
```

Verifique se sua versão da CLI do Azure é 2.18.0 ou superior:

```
az --version
```

Se precisar atualizar, tente o comando `az upgrade` (requer a versão 2.11+) ou confira [Instalar a CLI do Azure](#).

Em seguida, entre no Azure por meio da CLI:

```
az login
```

Esse comando abre um navegador para coletar suas credenciais. Quando o comando for concluído, ele mostrará a saída JSON que contém informações sobre as suas assinaturas.

Depois de conectado, você poderá executar os comandos do Azure com a CLI do Azure para trabalhar com

recursos na sua assinatura.

Está enfrentando problemas? [Fale conosco](#).

2. Clonar ou baixar o aplicativo de exemplo

- [Clone do Git](#)
- [Download](#)

Clone o repositório de exemplo:

```
git clone https://github.com/Azure-Samples/djangoapp
```

Em seguida, acesse esta pasta:

```
cd djangoapp
```

O exemplo djangoapp contém o aplicativo de enquetes do Django controlado por dados que você obtém seguindo [Escrever seu primeiro aplicativo Django](#) na documentação do Django. O aplicativo concluído é fornecido aqui para fins de conveniência.

O exemplo também é modificado para ser executado em um ambiente de produção, como o Serviço de Aplicativo:

- As configurações de produção estão no arquivo `azuresite/production.py`. Encontre as configurações de desenvolvimento em `azuresite/settings.py`.
- O aplicativo usa as configurações de produção quando a variável de ambiente `WEBSITE_HOSTNAME` é definida. O Serviço de Aplicativo do Azure define automaticamente essa variável como a URL do aplicativo Web, como `msdocs-django.azurewebsites.net`.

Essas configurações de produção são específicas para configurar o Django para execução em qualquer ambiente de produção e não são específicas do Serviço de Aplicativo. Para saber mais, confira a [Lista de verificação de implantação do Django](#). Confira também [Configurações de produção para o Django no Azure](#) para obter detalhes sobre algumas das alterações.

Está com problemas? [Fale conosco](#).

3. Criar um banco de dados Postgres no Azure

Instale a extensão `db-up` para a CLI do Azure:

```
az extension add --name db-up
```

Se o comando `az` não for reconhecido, verifique se você tem a CLI do Azure instalada, conforme descrito em [Configurar seu ambiente inicial](#).

Em seguida, crie o banco de dados Postgres no Azure com o comando `az postgres up`:

```
az postgres up --resource-group DjangoPostgres-tutorial-rg --location westus2 --sku-name B_Gen5_1 --server-name <postgres-server-name> --database-name pollsdb --admin-user <admin-username> --admin-password <admin-password> --ssl-enforcement Enabled
```

- Substitua `<postgres-server-name>` por um nome exclusivo em todo o Azure (o ponto de extremidade

do servidor se torna `https://<postgres-server-name>.postgres.database.azure.com`). Um bom padrão é usar uma combinação do nome da empresa e outro valor exclusivo.

- Em `<admin-username>` e `<admin-password>`, especifique as credenciais para criar um usuário administrador para esse servidor Postgres. O nome do usuário administrador não pode ser `azure_superuser`, `azure_pg_admin`, `admin`, `administrator`, `root`, `guest` nem `public`. Ele não pode começar com `pg_`. A senha precisa conter de **8 a 128 caracteres** de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números (0 a 9) e caracteres não alfanuméricos (por exemplo, !, # e %). A senha não pode conter o nome de usuário.
- Não use o caractere `$` no nome de usuário nem na senha. Posteriormente, você cria variáveis de ambiente com esses valores no qual o caractere `$` tem um significado especial dentro do contêiner do Linux usado para executar aplicativos Python.
- O **tipo de preço** `B_Gen5_1` (Básico, Geração 5, 1 núcleo) usado aqui é o mais barato. Para bancos de dados de produção, omita o argumento `--sku-name` para usar o nível `GP_Gen5_2` (Uso Geral, Geração 5, 2 núcleos) em vez disso.

Esse comando executa as seguintes ações, que podem levar alguns minutos:

- Criar um **grupo de recursos** chamado `DjangoPostgres-tutorial-rg` se ele não existir.
- Crie um servidor do Postgres nomeado de acordo com o argumento `--server-name`.
- Crie uma conta de administrador usando os argumentos `--admin-user` e `--admin-password`. Você pode omitir esses argumentos para permitir que o comando gere credenciais exclusivas.
- Crie um banco de dados `pollsdb`, nomeado de acordo com o argumento `--database-name`.
- Habilitar o acesso do endereço IP local.
- Habilitar o acesso dos serviços do Azure.
- Criando um usuário de banco de dados com acesso a um banco de dados `pollsdb`.

É possível executar todas as etapas separadamente com outros comandos `az postgres` e `psql`, mas `az postgres up` executa todas elas em conjunto.

Quando é concluído, o comando gera um objeto JSON que contém cadeias de conexão diferentes para o banco de dados, em conjunto com a URL do servidor, um nome de usuário gerado (como "joyfulKoala@msdocs-djangodb-12345") e uma senha de GUID. **Copie o nome de usuário e a senha para um arquivo de texto temporário**, pois você precisará deles mais tarde neste tutorial.

TIP

`-l <location-name>`, pode ser definido como qualquer uma das **regiões do Azure**. É possível obter as regiões disponíveis para sua assinatura com o comando `az account list-locations`. Para aplicativos de produção, coloque seu banco de dados e seu aplicativo na mesma localização.

Está com problemas? [Fale conosco](#).

4. Implantar o código no Serviço de Aplicativo do Azure

Nesta seção, você criará o host do aplicativo do Serviço de Aplicativo, conectará esse aplicativo ao banco de dados Postgres e implantará o código nesse host.

4.1 Criar o aplicativo do Serviço de Aplicativo

No terminal, verifique se você está na pasta do repositório `djangoapp` que contém o código do aplicativo.

Crie um aplicativo do Serviço de Aplicativo (o processo de host) com o comando `az webapp up`:

```
az webapp up --resource-group DjangoPostgres-tutorial-rg --location westus2 --plan DjangoPostgres-tutorial-plan --sku B1 --name <app-name>
```

- Para o argumento `--location`, use a mesma localização usado para o banco de dados na seção anterior.
- Substitua** `<app-name>` por um nome exclusivo em todo o Azure (o ponto de extremidade do servidor é `https://<app-name>.azurewebsites.net`). Os caracteres permitidos para `<app-name>` são A-Z, 0-9 e -. Um bom padrão é usar uma combinação do nome da empresa e um identificador de aplicativo.

Esse comando executa as seguintes ações, que podem levar alguns minutos:

- Criar o [grupo de recursos](#) se ele ainda não existir. (Neste comando, você usa o mesmo grupo de recursos no qual você criou o banco de dados anteriormente.)
- Criar o [plano do Serviço de Aplicativo](#) `DjangoPostgres-tutorial-plan` no tipo de preço Básico (B1) se ele não existir. `--plan` e `--sku` são opcionais.
- Criar o aplicativo do Serviço de Aplicativo se ele não existir.
- Habilitar o log padrão do aplicativo, se ainda não estiver habilitado.
- Carregar o repositório usando a implantação ZIP com a automação do build habilitada.
- Armazene em cache parâmetros comuns, como o nome do grupo de recursos e o Plano do Serviço de Aplicativo, no arquivo `.azure/config`. Como resultado, você não precisa especificar todos os mesmos parâmetros com comandos posteriores. Por exemplo, para reimplantar o aplicativo depois de fazer alterações, você pode simplesmente executar `az webapp up` novamente sem parâmetros. No entanto, os comandos provenientes de extensões da CLI, como `az postgres up`, não usam o cache no momento e, por isso, você precisou especificar o grupo de recursos e a localização aqui com o uso inicial de `az webapp up`.

Após a implantação bem-sucedida, o comando gera uma saída JSON como o seguinte exemplo:

```
{  
  "URL": "http://msdocs-djangoapp-12345.azurewebsites.net",  
  "appserviceplan": "DjangoPostgres-tutorial-plan",  
  "location": "westus",  
  "name": "msdocs-djangoapp-12345",  
  "os": "Linux",  
  "resourcegroup": "DjangoPostgres-tutorial-rg",  
  "runtime_version": "python|3.7",  
  "runtime_version_detected": "-.",  
  "sku": "BASIC",  
  "src_path": "//var//lib//postgresql//djangoapp"  
}
```

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

4.2 Configurar variáveis de ambiente para conexão com o banco de dados

Com o código implantado no Serviço de Aplicativo, a próxima etapa é conectar o aplicativo ao banco de dados Postgres no Azure.

O código do aplicativo espera encontrar informações sobre o banco de dados em quatro variáveis de ambiente chamadas `DBHOST`, `DBNAME`, `DBUSER` e `DBPASS`.

Para definir variáveis de ambiente no Serviço de Aplicativo, crie "configurações do aplicativo" usando o comando `az webapp config appsettings set` a seguir.

```
az webapp config appsettings set --settings DBHOST=<postgres-server-name> DBNAME="pollsdb" DBUSER=<username> DBPASS=<password>
```

- Substitua `<postgres-server-name>` pelo nome usado anteriormente com o comando `az postgres up`. O

código em `azuresite/production.py` acrescenta automaticamente `.postgres.database.azure.com` para criar a URL completa do servidor Postgres.

- Substitua `<username>` e `<password>` pelas credenciais de administrador que você usou com o comando `az postgres up` anterior ou por aquelas que `az postgres up` gerou para você. O código em `azuresite/production.py` constrói automaticamente o nome de usuário do Postgres completo com base em `DBUSER` e `DBHOST`, portanto, não inclua a parte `@server`. (Além disso, como observado anteriormente, você não deve usar o caractere `$` em nenhum valor, pois ele tem um significado especial para variáveis de ambiente do Linux.)
- O grupo de recursos e os nomes do aplicativo são extraídos dos valores armazenados em cache no arquivo `.azure/config`.

Em seu código Python, você acessa essas configurações como variáveis de ambiente com instruções como `os.environ.get('DBHOST')`. Para saber mais, confira [Acessar variáveis do ambiente](#).

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

4.3 Executar migrações de banco de dados do Django

As migrações de banco de dados do Django garantem que o esquema no PostgreSQL no banco de dados do Azure corresponda ao descrito em seu código.

1. Abra uma sessão SSH no **navegador** acessando a URL a seguir e entrando com suas credenciais de conta do Azure (não com as credenciais do servidor de banco de dados).

```
https://<app-name>.scm.azurewebsites.net/webssh/host
```

Substitua `<app-name>` pelo nome usado anteriormente no comando `az webapp up`.

Você também pode se conectar a uma sessão SSH com o comando `az webapp ssh`. No Windows, esse comando requer a CLI do Azure 2.18.0 ou superior.

Se você não puder se conectar à sessão SSH, o aplicativo não terá sido iniciado. Confira os logs de diagnóstico para obter detalhes. Por exemplo, se você não tiver criado as configurações de aplicativo necessárias na seção anterior, os logs indicarão `KeyError: 'DBNAME'`.

2. Na sessão SSH, execute os seguintes comandos (você pode colar os comandos usando **CTRL+Shift+V**):

```
# Run database migrations
python manage.py migrate

# Create the super user (follow prompts)
python manage.py createsuperuser
```

Se você encontrar erros relacionados à conexão com o banco de dados, verifique os valores das configurações do aplicativo criadas na seção anterior.

3. O comando `createsuperuser` solicita suas credenciais de superusuário. Para este tutorial, use o nome de usuário padrão `root`, pressione **Enter** para o endereço de email ser deixado em branco e digite `Pollsdb1` para a senha.
4. Se for exibido um erro informando que o banco de dados está bloqueado, verifique se você executou o comando `az webapp settings` na seção anterior. Sem essas configurações, o comando `migrate` não pode se comunicar com o banco de dados, resultando no erro.

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

4.4 Criar uma pergunta de enquete no aplicativo

1. Em um navegador, abra a URL `http://<app-name>.azurewebsites.net`. O aplicativo deve exibir a mensagem "Aplicativo de enquetes" e "Não há enquetes disponíveis" porque ainda não há enquetes específicas no banco de dados.

Se você vir "Erro de Aplicativo", é provável que não tenha criado as configurações necessárias na etapa anterior, [Configurar as variáveis de ambiente para conexão com o banco de dados](#) ou que esses valores contenham erros. Execute o comando `az webapp config appsettings list` para verificar as configurações. Você também pode [verificar os logs de diagnóstico](#) para ver erros específicos durante a inicialização do aplicativo. Por exemplo, se você não tiver criado as configurações, os logs mostrarão o erro

```
KeyError: 'DBNAME'
```

Após atualizar as configurações para corrigir os erros, aguarde um minuto para que o aplicativo seja reiniciado e atualize o navegador.

2. Navegue até `http://<app-name>.azurewebsites.net/admin`. Entre usando as credenciais de superusuário do Django da seção anterior (`root` e `Pollsdb1`). Em **Enquetes**, selecione **Adicionar** ao lado de **Perguntas** e crie uma enquete com algumas opções.
3. Navegue novamente para `http://<app-name>.azurewebsites.net` para confirmar que as perguntas agora são apresentadas ao usuário. Responda às perguntas como desejar para gerar dados no banco de dados.

Parabéns! Você está executando um aplicativo Web Django, escrito em Python, no Serviço de Aplicativo do Azure para Linux, com um banco de dados Postgres ativo.

Está com problemas? [Fale conosco](#).

NOTE

O Serviço de Aplicativo detecta um projeto Django procurando um arquivo `wsgi.py` em cada subpasta, que o `manage.py startproject` cria por padrão. Quando o Serviço de Aplicativo encontra o arquivo, ele carrega o aplicativo Web Django. Para obter mais informações, confira [Configurar imagem interna do Python](#).

5. Fazer alterações no código e reimplantar

Nesta seção, você faz alterações locais no aplicativo e reimplanta o código no Serviço de Aplicativo. No processo, você configura um ambiente virtual do Python que dá suporte ao trabalho em andamento.

5.1 Executar o aplicativo localmente

Em uma janela de terminal, execute os comandos a seguir. Siga os prompts ao criar o superusuário:

- [Bash](#)
- [PowerShell](#)
- [CMD](#)

```
# Configure the Python virtual environment
python3 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt
# Run Django migrations
python manage.py migrate
# Create Django superuser (follow prompts)
python manage.py createsuperuser
# Run the dev server
python manage.py runserver
```

Após o aplicativo Web ser totalmente carregado, o servidor de desenvolvimento do Django fornecerá a URL do aplicativo local na mensagem "Iniciando o servidor de desenvolvimento em <http://127.0.0.1:8000/>. Feche o servidor com CTRL-BREAK".

```
Performing system checks...

System check identified no issues (0 silenced).
June 24, 2020 - 10:27:14
Django version 2.2.13, using settings 'azuresite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Teste o aplicativo localmente com as seguintes etapas:

1. Acesse `http://localhost:8000` em um navegador, o que exibirá a mensagem "Não há sondagens disponíveis".
2. Acesse `http://localhost:8000/admin` e entre nele usando o usuário administrador criado anteriormente. Em **Enquetes**, selecione **Adicionar** novamente ao lado de **Perguntas** e crie uma enquete com algumas opções.
3. Vá para `http://localhost:8000` novamente e responda à pergunta para testar o aplicativo.
4. Pare o servidor Django pressionando **CTRL+C**.

Ao ser executado localmente, o aplicativo está usando um banco de dados SQLite3 local e não interfere no banco de dados de produção. Você também poderá usar um banco de dados PostgreSQL local, se desejar, para simular melhor seu ambiente de produção.

Está com problemas? [Fale conosco](#).

5.2 Atualizar o aplicativo

Em `polls/models.py`, localize a linha que começa com `choice_text` e altere o parâmetro `max_length` para 100:

```
# Find this line of code and set max_length to 100 instead of 200
choice_text = models.CharField(max_length=100)
```

Como você alterou o modelo de dados, crie uma migração do Django e migre o banco de dados:

```
python manage.py makemigrations
python manage.py migrate
```

Execute o servidor de desenvolvimento novamente com `python manage.py runserver` e teste o aplicativo em `http://localhost:8000/admin`:

Pare o servidor Web Django novamente com **CTRL +C**.

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

5.3 Reimplantar o código no Azure

Execute o seguinte comando na raiz do repositório:

```
az webapp up
```

Esse comando usa os parâmetros armazenados em cache no arquivo *.azure/config*. Como o Serviço de Aplicativo detecta que o aplicativo já existe, ele apenas reimplanta o código.

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

5.4 Executar migrações novamente no Azure

Como você fez alterações no modelo de dados, é necessário executar novamente as migrações de banco de dados no Serviço de Aplicativo.

Abra uma sessão SSH novamente no navegador acessando

<https://<app-name>.scm.azurewebsites.net/webssh/host> . Em seguida, execute o seguinte comando:

```
python manage.py migrate
```

Está enfrentando problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

5.5 Examinar o aplicativo na produção

Acesse <http://<app-name>.azurewebsites.net> e teste o aplicativo novamente em produção. (Como você alterou apenas o comprimento de um campo do banco de dados, a alteração só será perceptível se você tentar inserir uma resposta mais longa ao criar uma pergunta.)

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

6. Logs de diagnóstico de fluxo

Você pode acessar os logs do console gerados de dentro do contêiner que hospeda o aplicativo no Azure.

Execute o comando a seguir da CLI do Azure para ver o fluxo de logs. Esse comando usa parâmetros armazenados em cache no arquivo *.azure/config*.

```
az webapp log tail
```

Se você não vir os logs do console imediatamente, verifique novamente após 30 segundos.

Para interromper o streaming de log a qualquer momento, digite **Ctrl +C**.

Está com problemas? [Fale conosco](#).

NOTE

Você também pode inspecionar os arquivos de log do navegador em

```
https://<app-name>.scm.azurewebsites.net/api/logs/docker .
```

`az webapp up` ativa o log padrão para você. Por motivos de desempenho, esse log se desativa depois de algum tempo, mas volta é ativado novamente cada vez que você executa `az webapp up`. Para ativá-lo manualmente, execute o seguinte comando:

```
az webapp log config --docker-container-logging filesystem
```

7. Gerenciar seu aplicativo no portal do Azure

No [portal do Azure](#), pesquise pelo nome do aplicativo e selecione-o nos resultados.

The screenshot shows the Microsoft Azure portal interface. At the top, there is a search bar containing the text 'msdocs-django'. Below the search bar, the results are displayed under the 'Resources' section. A blue box highlights the first result, 'msdocs-djangoapp-12345', which is listed under the 'App Service' category. Another result, 'msdocs-djangodb-12345', is listed under 'Azure Database for PostgreSQL server'. To the left of the search results, there is a sidebar with a 'Create a resource' button and a 'Recent resources' section.

Por padrão, o portal mostra a página de **Visão geral** do aplicativo, que fornece uma exibição do desempenho geral. Aqui você também pode executar tarefas básicas de gerenciamento como procurar, parar, reiniciar e excluir. As guias no lado esquerdo da página mostram as páginas de configuração diferentes que você pode abrir.

The screenshot shows the Microsoft Azure portal with the URL 'https://msdocs-djangoapp-12345.azurewebsites.net' in the address bar. The main area is titled 'msdocs-djangoapp-12345' and shows the 'App Service' configuration. On the left, there is a navigation menu with sections like 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Security', 'Events (preview)', 'Deployment', 'Quickstart', 'Deployment slots', 'Deployment Center', 'Settings', 'Configuration', 'Authentication / Authorization', 'Application Insights', and 'Identity'. The 'Overview' tab is selected. In the center, there is a table with details such as 'Resource group (change) : DjangoPostgres-tutorial-rg', 'Status : Running', 'Location : West US 2', 'Subscription (change) : Primary', 'Subscription ID : 44f7aaf4-97f8-4aa9-a791', and 'Tags (change) : Click here to add tags'. Below the table, there are two cards: 'Diagnose and solve problems' and 'App Service Advisor'. At the bottom, there are three charts: 'Http 5xx', 'Data In', and 'Data Out'.

Está com problemas? Veja primeiro o [Guia de solução de problemas](#). Caso contrário, [fale conosco](#).

8. Limpar os recursos

Se quiser manter o aplicativo ou prosseguir para os tutoriais adicionais, pule para as [Próximas etapas](#). Caso contrário, para evitar incorrer em encargos contínuos, você pode excluir o grupo de recursos criado para este tutorial:

```
az group delete --no-wait
```

O comando usa o nome do grupo de recursos armazenado em cache no arquivo `.azure/config`. Ao excluir o grupo de recursos, você também desaloca e exclui todos os recursos contidos nele.

A exclusão de todos os recursos pode levar algum tempo. O argumento `--no-wait` permite que o comando seja retornado imediatamente.

Está com problemas? [Fale conosco](#).

Próximas etapas

Saiba como mapear um nome DNS personalizado para seu aplicativo:

[Tutorial: Mapear o nome DNS personalizado para seu aplicativo](#)

Saiba como o Serviço de Aplicativo executa um aplicativo Python:

[Configurar o aplicativo Python](#)

Tutorial: Monitorar e ajustar o Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 3 minutes to read

Banco de Dados do Azure para PostgreSQL tem recursos que ajudam você a compreender e melhorar o desempenho do servidor. Neste tutorial, você aprenderá a:

- Habilitar a consulta e aguardar a coleta de estatísticas
- Acessar e utilizar os dados coletados
- Exibir desempenho de consulta e estatísticas de espera ao longo do tempo
- Analisar um banco de dados para obter as recomendações de desempenho
- Aplicar recomendações do desempenho

Pré-requisitos

Você precisa de um Banco de Dados do Azure para PostgreSQL versão 9.6 ou 10. Você pode seguir as etapas no [Tutorial de criação](#) para criar um servidor.

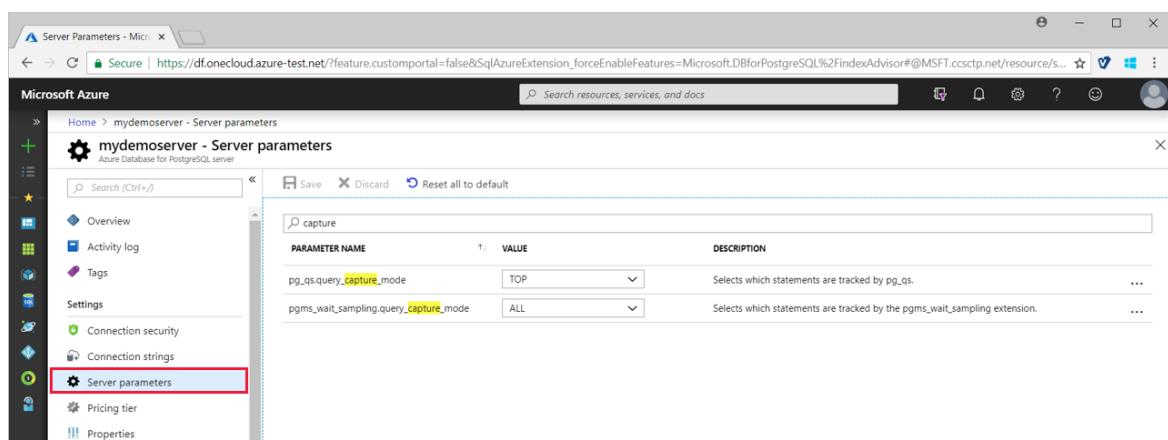
IMPORTANT

Repositório de Consultas, Análise de Desempenho de Consultas, e Recomendação de Desempenho estão em Visualização Pública.

Habilitar coleta de dados

O [Repositório de Consultas](#) captura um histórico das consultas e estatísticas de espera em seu servidor e os armazena no banco de dados `azure_sys` em seu servidor. É um recurso opcional. Para habilitá-lo:

1. Abra o portal do Azure.
2. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL.
3. Selecionar **Servidor de parâmetros** que estiver nas seções de **Configurações** do menu à esquerda.
4. Definir `pg_qs.query_capture_mode` a **superior** para começar a coletar dados de desempenho de consulta. Definir `pgms_wait_sampling.query_capture_mode` à **todos** os para iniciar a coltar as estatísticas de espera. Salve.

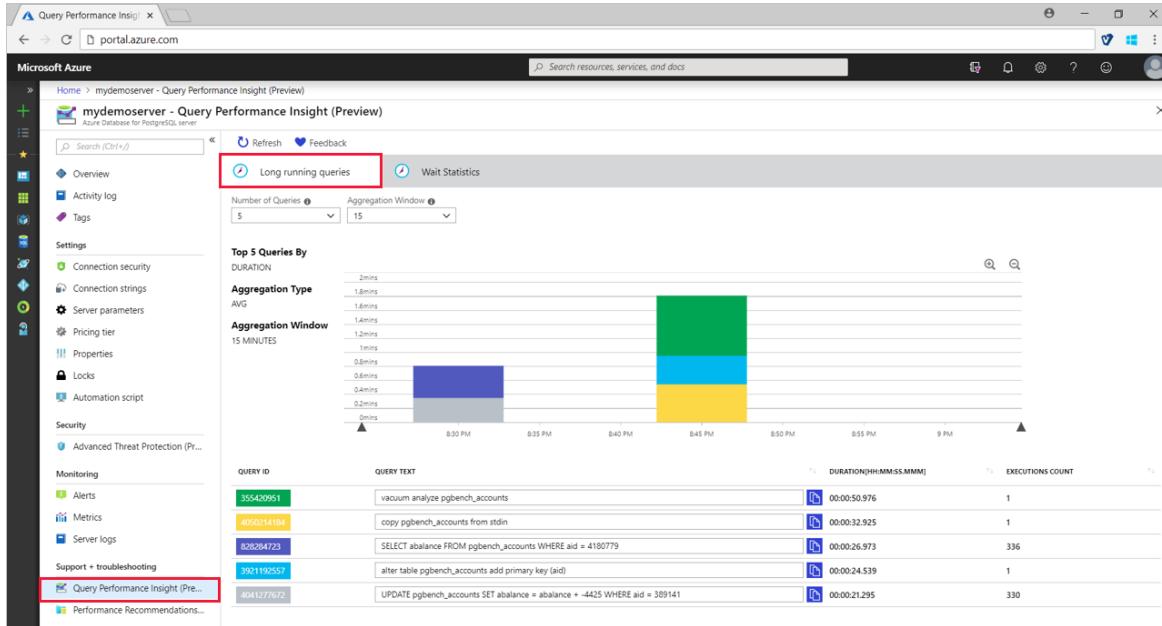


- Permita que o primeiro lote de dados persista no banco de dados `azure_sys` por até 20 minutos.

Insights de desempenho

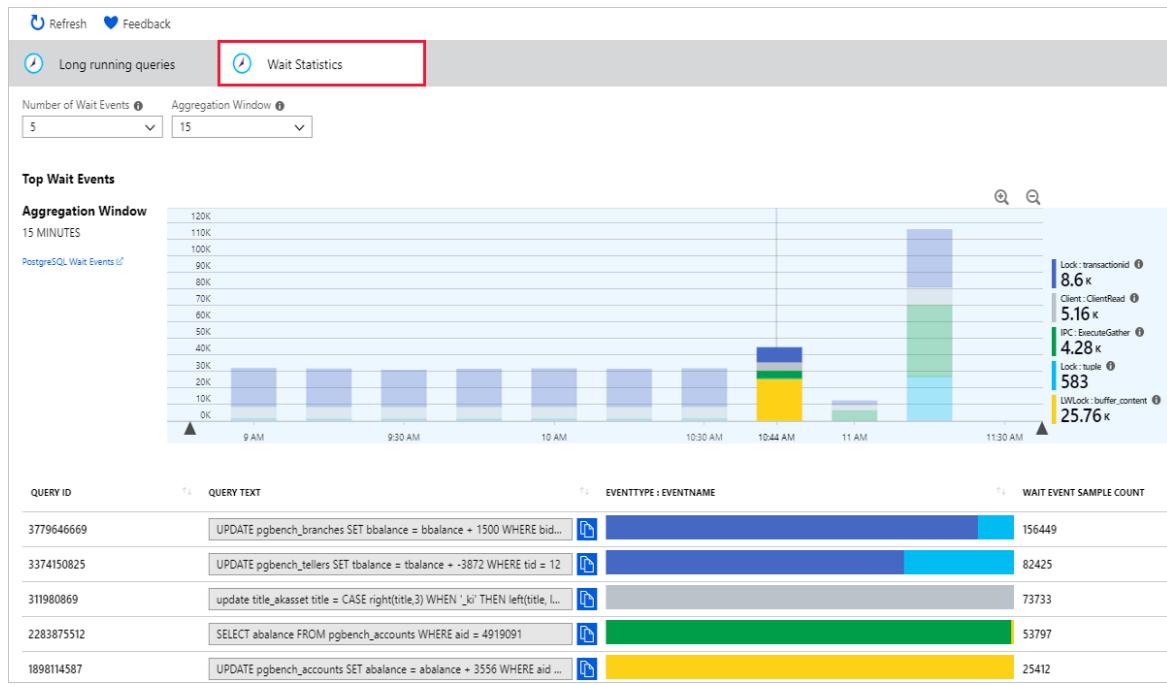
A visualização da [Análise de Desempenho de Consultas](#) no portal do Azure será superficial visualizações em informações do Repositório de Consultas.

- Na página do portal do Banco de Dados do Azure para PostgreSQL, selecione **Insight do Desempenho de Consulta** em suporte + seção solução de problemas do menu à esquerda.
- A guia **consultas de Longa execução** mostra as 5 consultas superiores por duração média por execução, agregadas em intervalos de 15 minutos.



Você pode exibir mais consultas, selecionando a partir do **Número de consultas** lista suspensa. As cores do gráfico pode ser alteradas para uma ID de consulta específica ao fazer isso.

- Você pode clicar e arrastar no gráfico para restringi-lo a uma janela de tempo específico.
- Use os ícones de ampliar e afastar para exibir um período maior ou menor, respectivamente.
- Exiba a tabela abaixo do gráfico para obter mais detalhes sobre as consultas de longa execução na janela de tempo.
- Selecione a guia das **Estatísticas de Espera** guia para exibir as visualizações correspondentes em espera no servidor.



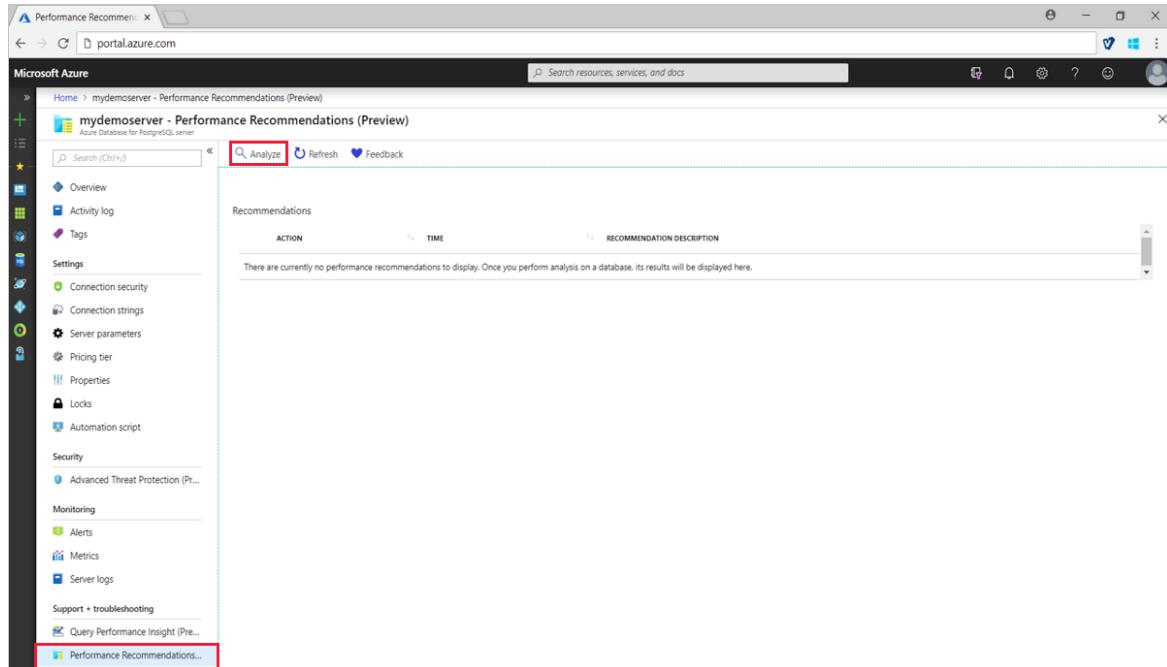
Permissões

Permissões do Proprietário ou Colaborador necessárias para exibir o texto das consultas na Análise de Desempenho de Consultas Leitor podem exibir gráficos e tabelas, mas não o texto da consulta.

Recomendações do desempenho

O recurso das [Recomendações de Desempenho](#) recurso analisa as cargas de trabalho entre seu servidor para identificar os índices com o potencial de melhorar o desempenho.

1. Abra **Recomendações de Desempenho** da seção suporte + solução de problemas da barra de menus, na página do portal do Azure para seu servidor PostgreSQL.



2. Selecione **Analisar** e escolha um banco de dados. Isso iniciará a análise.
3. Dependendo de sua carga de trabalho, isso pode levar vários minutos para ser concluído. Quando a análise for concluída, haverá uma notificação no portal.
4. A janela **Recomendações de Desempenho** Mostrará uma lista de recomendações, caso seja encontrada.

5. Uma recomendação mostrará informações sobre os relevantes banco de dados, tabela, coluna, e tamanho de índice.

The screenshot shows the 'Recommendations' section of the Azure Database Performance Advisor. It lists a single recommendation:

ACTION	TIME	RECOMMENDATION DESCRIPTION
CREATE INDEX	09/12/2018, 2:17:03 PM	Database: pgbench Table: pgbench_accounts Columns: aid Estimated Index Size: 26.8125 MB Command: create index on "public"."pgbench_accounts"(aid);

6. Para implementar a recomendação, copie o texto da consulta e executá-lo do seu cliente de escolha.

Permissões

Permissões de Proprietário ou Colaborador necessárias para executar a análise usando o recurso de recomendações de desempenho.

Limpar os recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão *Excluir* na página *Visão geral* do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão *Excluir* final.

Próximas etapas

Saiba mais sobre [monitoramento e ajuste](#) no Banco de Dados do Azure para PostgreSQL.

Exemplos da CLI do Azure para o Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 2 minutes to read

A tabela a seguir inclui links para exemplos de scripts da CLI do Azure para o Banco de Dados do Azure para PostgreSQL.

LINK DE EXEMPLO	DESCRIÇÃO
Criar um servidor	
Criar uma regra de firewall e servidor	O script da CLI do Azure que cria um servidor de Banco de Dados do Azure para PostgreSQL e configura uma regra de firewall no nível do servidor.
Dimensionar um servidor	
Dimensionar um servidor	Script da CLI do Azure que amplia ou reduz um servidor do Banco de Dados do Azure para PostgreSQL para permitir alterações nas necessidades de desempenho.
Alterar as configurações do servidor	
Alterar as configurações do servidor	Script da CLI do Azure que altera opções de configuração de um servidor de Banco de Dados do Azure para PostgreSQL.
Restaurar um servidor	
Restaurar um servidor	Script CLI do Azure que restaura um Banco de Dados do Azure para PostgreSQL a um ponto anterior.
Baixar os logs do servidor	
Habilitar e fazer download dos logs do servidor	Script da CLI do Azure que habilita e baixa logs do servidor de um Banco de Dados do Azure para PostgreSQL.

Definições internas do Azure Policy para o Banco de Dados do Azure para PostgreSQL

16/06/2021 • 5 minutes to read

Esta página é um índice de definições de políticas internas do [Azure Policy](#) para o Banco de Dados do Azure para PostgreSQL. Para obter políticas internas adicionais do Azure Policy para outros serviços, confira [Definições internas do Azure Policy](#).

O nome de cada definição de política interna leva à definição da política no portal do Azure. Use o link na coluna **Versão** para exibir a origem no [repositório GitHub do Azure Policy](#).

Banco de Dados do Azure para PostgreSQL

NOME (PORTAL DO AZURE)	DESCRIÇÃO	EFEITO(S)	VERSÃO (GITHUB)
Configurar a Proteção Avançada contra Ameaças como habilitada no banco de dados do Azure para servidores PostgreSQL	Habilite a Proteção Avançada contra Ameaças nos seus bancos de dados do Azure de nível Não Básico para que os servidores PostgreSQL detectem atividades anômalas que indiquem tentativas incomuns e possivelmente prejudiciais no sentido de acessar ou explorar bancos de dados.	DeployIfNotExists, desabilitado	1.0.0
A limitação de conexão deve estar habilitada para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a limitação de conexão habilitada. Essa configuração permite a otimização temporária da conexão por IP para muitas falhas inválidas de login de senha.	AuditIfNotExists, desabilitado	1.0.0
As desconexões devem ser registradas em log para os servidores de banco de dados PostgreSQL.	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem log_disconnections habilitada.	AuditIfNotExists, desabilitado	1.0.0

NOME	DESCRIÇÃO	EFEITO(S)	VERSÃO
'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	<p>O Banco de Dados do Azure para PostgreSQL dá suporte à conexão de seu servidor de Banco de Dados do Azure para PostgreSQL para aplicativos cliente usando o protocolo SSL.</p> <p>Impor conexões SSL entre seu servidor de banco de dados e os aplicativos cliente ajuda a proteger contra ataques de "intermediários" criptografando o fluxo de dados entre o servidor e seu aplicativo. Essa configuração impõe que o SSL sempre esteja habilitado para acessar seu servidor de banco de dados.</p>	Audit, desabilitado	1.0.1
O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	<p>O Banco de Dados do Azure para PostgreSQL permite que você escolha a opção de redundância para seu servidor de banco de dados. Ele pode ser definido como um armazenamento de backup com redundância geográfica no qual os dados não são armazenados apenas na região em que o servidor está hospedado, mas também é replicado para uma região emparelhada para dar a opção de recuperação em caso de falha de região. A configuração do armazenamento com redundância geográfica para backup só é permitida durante a criação do servidor.</p>	Audit, desabilitado	1.0.1

NOME	DESCRÍÇÃO	EFEITO(S)	VERSÃO
A criptografia de infraestrutura deve ser habilitada para servidores do Banco de Dados do Azure para PostgreSQL	Habilite a criptografia de infraestrutura para os servidores do Banco de Dados do Azure para PostgreSQL terem um nível mais alto de garantia de segurança dos dados. Quando a criptografia de infraestrutura está habilitada, os dados em repouso são criptografados duas vezes usando chaves gerenciadas da Microsoft em conformidade com o FIPS 140-2	Audit, Deny, desabilitado	1.0.0
Os pontos de verificação de log devem ser habilitados para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a configuração log_checkpoints habilitada.	AuditIfNotExists, desabilitado	1.0.0
As conexões de log devem ser habilitadas para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a configuração log_connections habilitada.	AuditIfNotExists, desabilitado	1.0.0
A duração do log deve ser habilitada para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a configuração log_duration habilitada.	AuditIfNotExists, desabilitado	1.0.0
O servidor PostgreSQL deve usar um ponto de extremidade de serviço de rede virtual	As regras de firewall baseadas em rede virtual são usadas para habilitar o tráfego de uma sub-rede específica para o Banco de Dados do Azure para PostgreSQL e ainda garantir que o tráfego permaneça dentro do limite do Azure. Essa política oferece uma forma de auditar se o Banco de Dados do Azure para PostgreSQL tem um ponto de extremidade de serviço de rede virtual em uso.	AuditIfNotExists, desabilitado	1.0.2

NOME	DESCRIÇÃO	EFEITO(S)	VERSÃO
Os servidores PostgreSQL devem usar chaves gerenciadas pelo cliente para criptografar dados inativos	<p>Use chaves gerenciadas pelo cliente para gerenciar a criptografia em repouso de seus servidores PostgreSQL. Por padrão, os dados são criptografados em repouso com chaves gerenciadas pelo serviço, mas as chaves gerenciadas pelo cliente normalmente são necessárias para atender aos padrões de conformidade regulatória. As chaves gerenciadas pelo cliente permitem que os dados sejam criptografados com uma chave do Azure Key Vault criada por você e de sua propriedade. Você tem total controle e responsabilidade pelo ciclo de vida da chave, incluindo rotação e gerenciamento.</p>	AuditIfNotExists, desabilitado	1.0.4
O ponto de extremidade privado deve ser habilitado para servidores PostgreSQL	<p>As conexões de ponto de extremidade privado impõem comunicações seguras habilitando a conectividade privada ao Banco de Dados do Azure para PostgreSQL. Configure uma conexão de ponto de extremidade privado para habilitar o acesso ao tráfego proveniente somente de redes conhecidas e impedir o acesso de todos os outros endereços IP, incluindo no Azure.</p>	AuditIfNotExists, desabilitado	1.0.2
O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	<p>Desabilitar a propriedade de acesso à rede pública aprimora a segurança garantindo que o Banco de Dados do Azure para servidores flexíveis do PostgreSQL só possa ser acessado de um ponto de extremidade privado. Essa configuração desabilita estritamente o acesso de qualquer espaço de endereço público fora do intervalo de IP do Azure e nega todos os logons que correspondam a regras de firewall baseadas em IP ou em rede virtual.</p>	Audit, Deny, desabilitado	1.0.0

NOME	DESCRIÇÃO	EFEITO(S)	VERSAO
O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	Desabilitar a propriedade de acesso à rede pública aprimora a segurança e garantir que o Banco de Dados do Azure para PostgreSQL só possa ser acessado de um ponto de extremidade privado. Essa configuração desabilita o acesso de qualquer espaço de endereço público fora do intervalo de IP do Azure e nega todos os logons que correspondam a regras de firewall baseadas em IP ou em rede virtual.	Audit, desabilitado	1.0.2

Próximas etapas

- Confira os internos no [repositório Azure Policy GitHub](#).
- Revise a [estrutura de definição do Azure Policy](#).
- Revisar [Compreendendo os efeitos da política](#).

Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 4 minutes to read

Este artigo apresenta diretrizes e considerações para trabalhar com o Banco de Dados do Azure para PostgreSQL – Servidor único.

O que é um servidor do Banco de Dados do Azure para PostgreSQL?

Um servidor na opção de implantação de Banco de Dados do Azure para PostgreSQL – Servidor único é um ponto administrativo central para vários bancos de dados. É a mesma construção de servidor PostgreSQL com a qual talvez você já esteja familiarizado no mundo local. Especificamente, o serviço PostgreSQL é gerenciado, oferece garantias de desempenho, expõe acesso e recursos no nível do servidor.

Um Banco de Dados do Azure para servidor PostgreSQL:

- É criado dentro de uma assinatura do Azure.
- É o recurso pai para bancos de dados.
- Fornece um namespace para bancos de dados.
- É um contêiner com semântica de tempo de vida fortes – exclua um servidor e ele excluirá os bancos de dados contidos.
- Coloca recursos em uma região.
- Fornece um ponto de extremidade de conexão para acesso ao servidor e ao banco de dados
- Fornece o escopo para políticas de gerenciamento que se aplicam a seus bancos de dados: logons, firewall, usuários, funções, configurações etc.
- Está disponível em várias versões. Para saber mais, confira [Versões do banco de dados PostgreSQL com suporte](#).
- É extensível pelos usuários. Para saber mais, confira [Extensões do PostgreSQL](#).

Dentro de um banco de dados do Azure para o servidor PostgreSQL, você pode criar um ou mais bancos de dados. Você pode optar por criar um banco de dados por servidor para utilizar todos os recursos ou criar vários bancos de dados para compartilhar os recursos. Os preços são estruturados por servidor, com base na configuração do tipo de preço, vCores e armazenamento (GB). Para obter mais informações, consulte [Tipos de preço](#).

Como faço para me conectar e autenticar em um Banco de Dados do Azure para servidor PostgreSQL?

Os elementos a seguir ajudam a garantir o acesso seguro ao seu banco de dados:

CONCEITO DE SEGURANÇA	DESCRÍÇÃO
Autenticação e autorização	O Banco de Dados do Azure para servidor PostgreSQL oferece suporte à autenticação de PostgreSQL nativa. Você pode se conectar e autenticar no servidor com logon de administrador do servidor.

CONCEITO DE SEGURANÇA	DESCRIÇÃO
Protocolo	O serviço oferece suporte a um protocolo baseado em mensagem usado pelo PostgreSQL.
TCP/IP	O protocolo tem suporte em TCP/IP e em soquetes de domínio do Unix.
Firewall	Para ajudar a proteger seus dados, uma regra de firewall impede todo acesso ao servidor e seus bancos de dados até que você especifique quais computadores têm permissão. Confira Regras de firewall do Banco de Dados do Azure para servidor PostgreSQL .

Gerenciando o servidor

Você pode gerenciar o Banco de Dados do Azure para servidores PostgreSQL usando o [Portal do Azure](#) ou a [CLI do Azure](#).

Ao criar um servidor, você configura as credenciais de seu usuário administrador. O usuário administrador é o usuário com privilégio mais elevado no servidor. Ele pertence à função `azure_pg_admin`. Essa função não tem permissões completas de superusuário.

O atributo de superusuário do PostgreSQL é atribuído a `azure_superuser`, que pertence ao serviço gerenciado. Você não tem acesso a essa função.

Um Banco de Dados do Azure para PostgreSQL possui bancos de dados padrão:

- **postgres** – um banco de dados padrão a que você poderá se conectar após seu servidor ser criado.
- **azure_maintenance** – este banco de dados é usado para separar os processos que fornecem o serviço gerenciado das ações do usuário. Você não tem acesso a esse banco de dados.
- **azure_sys** – um banco de dados para o Repositório de Consultas. Este banco de dados não acumula dados quando o Repositório de Consultas está desativado; essa é a configuração padrão. Para obter mais informações, confira o tópico [Visão geral do Repositório de Consultas](#).

Parâmetros do Servidor

Os parâmetros de servidor PostgreSQL determinam a configuração do servidor. No Banco de Dados do Azure para PostgreSQL, a lista de parâmetros pode ser exibida e editada usando o portal do Azure ou a CLI do Azure.

Como um serviço gerenciado para Postgres, os parâmetros configuráveis no banco de dados do Azure para PostgreSQL são um subconjunto dos parâmetros em uma instância local do Postgres (para obter mais informações sobre parâmetros Postgres, consulte o [PostgreSQL documentação](#)). O banco de dados do Azure para servidor PostgreSQL está habilitado com valores padrão para cada parâmetro na criação. Alguns parâmetros que necessitariam de um reinício de servidor ou de acesso de superusuário para as mudanças terem efeito não podem ser configurados pelo usuário.

Próximas etapas

- Para obter uma visão geral do serviço, confira [Visão geral do Banco de Dados para PostgreSQL](#).
- Para saber mais sobre cotas e limitações específicas de recursos com base em sua **camada de serviço**, confira [Camadas de serviço](#).
- Para saber mais sobre como se conectar ao serviço, confira [Bibliotecas de conexão para o Banco de Dados do Azure para PostgreSQL](#).
- Exibir e editar os parâmetros de servidor por meio de [portal do Azure](#) ou [CLI do Azure](#).

Versões principais do PostgreSQL compatíveis

21/05/2021 • 2 minutes to read

Confira a [política de controle de versão do Banco de Dados do Azure para PostgreSQL](#) para obter detalhes da política de suporte.

No momento, o Banco de Dados do Azure para PostgreSQL dá suporte às seguintes versões principais:

PostgreSQL versão 11

A versão secundária atual é 11.6. Veja a [documentação do PostgreSQL](#) para saber mais sobre os aprimoramentos e as correções nesta versão secundária.

PostgreSQL versão 10

A versão secundária atual é 10.11. Veja a [documentação do PostgreSQL](#) para saber mais sobre os aprimoramentos e as correções nesta versão secundária.

PostgreSQL versão 9.6

A versão secundária atual é 9.6.16. Veja a [documentação do PostgreSQL](#) para saber mais sobre os aprimoramentos e as correções nesta versão secundária.

PostgreSQL versão 9.5 (desativada)

Alinhando-se com a [política de controle de versão](#) da comunidade do Postgres, o Banco de Dados do Azure para PostgreSQL desativou o Postgres versão 9.5 desde 11 de fevereiro de 2021. Confira a [política de controle de versão do Banco de Dados do Azure para PostgreSQL](#) para obter mais detalhes e restrições. Se estiver executando esta versão principal, atualize-a para uma versão posterior, preferencialmente, para o PostgreSQL 11 assim que possível.

Como gerenciar atualizações

O projeto PostgreSQL emite regularmente versões secundárias para corrigir bugs relatados. O Banco de Dados do Azure para PostgreSQL corrige automaticamente os servidores com versões secundárias durante as implantações mensais do serviço.

Não há suporte para atualizações automáticas in-loco em versões principais. Para fazer a atualização para uma versão principal posterior, você poderá

- Usar um dos métodos documentados em [Atualizações da versão principal usando o despejo e a restauração](#).
- Usar `pg_dump` e `pg_restore` para mover um banco de dados para um servidor criado com a nova versão do mecanismo.
- Usar o [Serviço de Migração de Banco de Dados do Azure](#) para fazer atualizações online.

Sintaxe da versão

Antes do PostgreSQL versão 10, a [política de controle de versão do PostgreSQL](#) considerava uma atualização de *versão principal* como sendo um aumento no primeiro *ou* no segundo número. Por exemplo, 9.5 para 9.6 era considerado uma atualização de versão *principal*. Na versão 10 em diante, apenas uma alteração no primeiro número é considerada uma atualização de versão principal. Por exemplo, 10.0 para 10.1 é uma atualização de versão *secundária*. A versão 10 para 11 é uma atualização de versão *principal*.

Próximas etapas

Para obter informações sobre as extensões PostgreSQL compatíveis, confira [o documento de extensões](#).

Extensões PostgreSQL no Banco de Dados do Azure para PostgreSQL – Servidor único

09/08/2021 • 13 minutes to read

O PostgreSQL fornece a capacidade de estender a funcionalidade de seu banco de dados usando as extensões. As extensões agrupam vários objetos SQL em um pacote que pode ser carregado ou removido do seu banco de dados com um comando. Depois de carregadas no banco de dados, as extensões funcionam como recursos internos.

Como usar as extensões PostgreSQL

As extensões PostgreSQL devem ser instaladas no banco de dados para que você possa usá-las. Para instalar uma extensão específica, execute o comando `CREATE EXTENSION` na ferramenta psql para carregar os objetos empacotados em seu banco de dados.

O Banco de Dados do Azure para PostgreSQL dá suporte a um subconjunto de extensões de chave, conforme listado abaixo. Essas informações também estão disponíveis por meio da execução de

`SELECT * FROM pg_available_extensions;`. Não há suporte para as extensões além daquelas listadas. Você não pode criar uma extensão própria no Banco de Dados do Azure para PostgreSQL.

Extensões do Postgres 11

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL que têm a versão 11 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	2.5.1	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	2.5.1	Exemplo de conjunto de dados de padronizador de endereço dos EUA
btree_gin	1.3	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1.5	suporte para indexação de tipos de dados comuns no GiST
citext	1.5	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.4	tipo de dados para cubos multidimensionais
dblink	1.2	conecta-se a outros bancos de dados PostgreSQL em um banco de dados

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
earthdistance	1.1	calcula distâncias de grande círculo na superfície da Terra
fuzzystrmatch	1.1	determina as semelhanças e a distância entre cadeias de caracteres
hstore	1.5	tipo de dados para armazenar conjuntos de pares (chave, valor)
hypopg	1.1.2	Índices hipotéticos para o PostgreSQL
intarray	1.2	suporte a funções, operadores e índice para matrizes 1D de inteiros
isn	1.2	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.1	tipo de dados para estruturas semelhantes a árvores hierárquicas
orafce	3.7	Funções e operadores que emulam um subconjunto de funções e pacotes do RDBMS comercial
pgaudit	1.3.1	fornecer a funcionalidade de auditoria
pgcrypto	1.3	funções criptográficas
pgrouting	2.6.2	Extensão de pgRouting
pgrowlocks	1.2	mostra informações de bloqueio no nível de linha
pgstattuple	1.5	mostra estatísticas no nível da tupla
pg_buffercache	1.3	examina o cache do buffer compartilhado
pg_partman	4.0.0	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1.2	pré-aquecer dados de relações
pg_stat_statements	1.6	acompanhar as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1.4	medição de similaridade de texto e da pesquisa de índice com base em trigramas

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
plv8	2.3.11	Linguagem de procedimento confiável PL/JavaScript (v8)
postgis	2.5.1	Funções e tipos espaciais de geometria, geografia e rasterização do PostGIS
postgis_sfcgal	2.5.1	Funções SFCGAL do PostGIS
postgis_tiger_geocoder	2.5.1	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	2.5.1	Tipos e funções espaciais da topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
tablefunc	1.0	funções que manipulam tabelas inteiras, incluindo a tabela de referência cruzada
timescaledb	1.7.4	Permite inserções escalonáveis e consultas complexas em dados de série temporal
unaccent	1.1	dicionário de pesquisa de texto que remove sotaques
uuid-ossp	1.1	gera UUIDs (identificadores universais exclusivos)

Extensões do Postgres 10

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL que têm a versão 10 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	2.5.1	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	2.5.1	Exemplo de conjunto de dados de padronizador de endereço dos EUA
btree_gin	1.3	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1.5	suporte para indexação de tipos de dados comuns no GiST

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
chkpass	1.0	tipo de dados para senhas criptografadas automaticamente
citext	1.4	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.2	tipo de dados para cubos multidimensionais
dblink	1.2	conecta-se a outros bancos de dados PostgreSQL em um banco de dados
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
earthdistance	1.1	calcula distâncias de grande círculo na superfície da Terra
fuzzystrmatch	1.1	determina as semelhanças e a distância entre cadeias de caracteres
hstore	1.4	tipo de dados para armazenar conjuntos de pares (chave, valor)
hypopg	1.1.1	Índices hipotéticos para o PostgreSQL
intarray	1.2	suporte a funções, operadores e índice para matrizes 1D de inteiros
isn	1.1	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.1	tipo de dados para estruturas semelhantes a árvores hierárquicas
orafce	3.7	Funções e operadores que emulam um subconjunto de funções e pacotes do RDBMS comercial
pgaudit	1.2	fornece a funcionalidade de auditoria
pgcrypto	1.3	funções criptográficas
pgrouting	2.5.2	Extensão de pgRouting
pgrowlocks	1.2	mostra informações de bloqueio no nível de linha
pgstattuple	1.5	mostra estatísticas no nível da tupla
pg_buffercache	1.3	examina o cache do buffer compartilhado

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
pg_partman	2.6.3	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1,1	pré-aquecer dados de relações
pg_stat_statements	1.6	acompanhar as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1,3	medição de similaridade de texto e da pesquisa de índice com base em trigramas
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
plv8	2.1.0	Linguagem de procedimento confiável PL/JavaScript (v8)
postgis	2.4.3	Funções e tipos espaciais de geometria, geografia e rasterização do PostGIS
postgis_sfcgal	2.4.3	Funções SFCGAL do PostGIS
postgis_tiger_geocoder	2.4.3	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	2.4.3	Tipos e funções espaciais da topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
tablefunc	1.0	funções que manipulam tabelas inteiras, incluindo a tabela de referência cruzada
timescaledb	1.7.4	Permite inserções escalonáveis e consultas complexas em dados de série temporal
unaccent	1,1	dicionário de pesquisa de texto que remove sotaques
uuid-ossp	1,1	gera UUIDs (identificadores universais exclusivos)

Extensões do Postgres 9.6

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL que têm a versão 9.6 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	2.3.2	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	2.3.2	Exemplo de conjunto de dados de padronizador de endereço dos EUA
btree_gin	1.0	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1.2	suporte para indexação de tipos de dados comuns no GiST
chkpass	1.0	tipo de dados para senhas criptografadas automaticamente
citext	1.3	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.2	tipo de dados para cubos multidimensionais
dblink	1.2	conecta-se a outros bancos de dados PostgreSQL em um banco de dados
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
earthdistance	1.1	calcula distâncias de grande círculo na superfície da Terra
fuzzystrmatch	1.1	determina as semelhanças e a distância entre cadeias de caracteres
hstore	1.4	tipo de dados para armazenar conjuntos de pares (chave, valor)
hypopg	1.1.1	Índices hipotéticos para o PostgreSQL
intarray	1.2	suporte a funções, operadores e índice para matrizes 1D de inteiros
isn	1.1	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.1	tipo de dados para estruturas semelhantes a árvores hierárquicas
orafce	3.7	Funções e operadores que emulam um subconjunto de funções e pacotes do RDBMS comercial
pgaudit	1.1.2	fornecere a funcionalidade de auditoria

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
pgcrypto	1.3	funções criptográficas
pgrouting	2.3.2	Extensão de pgRouting
pgrowlocks	1.2	mostra informações de bloqueio no nível de linha
pgstattuple	1.4	mostra estatísticas no nível da tupla
pg_buffercache	1.2	examina o cache do buffer compartilhado
pg_partman	2.6.3	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1.1	pré-aquecer dados de relações
pg_stat_statements	1.4	acompanhar as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1.3	medição de similaridade de texto e da pesquisa de índice com base em trigramas
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
plv8	2.1.0	Linguagem de procedimento confiável PL/JavaScript (v8)
postgis	2.3.2	Funções e tipos espaciais de geometria, geografia e rasterização do PostGIS
postgis_sfcgal	2.3.2	Funções SFCGAL do PostGIS
postgis_tiger_geocoder	2.3.2	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	2.3.2	Tipos e funções espaciais da topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
tablefunc	1.0	funções que manipulam tabelas inteiras, incluindo a tabela de referência cruzada
timescaledb	1.7.4	Permite inserções escalonáveis e consultas complexas em dados de série temporal

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
unaccent	1,1	dicionário de pesquisa de texto que remove sotaques
uuid-ossp	1,1	gera UUIDs (identificadores universais exclusivos)

Extensões do Postgres 9.5

NOTE

O PostgreSQL versão 9.5 foi desativado.

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL que têm a versão 9.5 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	2.3.0	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	2.3.0	Exemplo de conjunto de dados de padronizador de endereço dos EUA
btree_gin	1.0	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1,1	suporte para indexação de tipos de dados comuns no GiST
chkpass	1.0	tipo de dados para senhas criptografadas automaticamente
citext	1,1	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.0	tipo de dados para cubos multidimensionais
dblink	1,1	conecta-se a outros bancos de dados PostgreSQL em um banco de dados
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
earthdistance	1.0	calcula distâncias de grande círculo na superfície da Terra
fuzzystrmatch	1.0	determina as semelhanças e a distância entre cadeias de caracteres

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
hstore	1,3	tipo de dados para armazenar conjuntos de pares (chave, valor)
hypopg	1.1.1	Índices hipotéticos para o PostgreSQL
intarray	1.0	suporte a funções, operadores e índice para matrizes 1D de inteiros
isn	1.0	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.0	tipo de dados para estruturas semelhantes a árvores hierárquicas
orafce	3.7	Funções e operadores que emulam um subconjunto de funções e pacotes do RDBMS comercial
pgaudit	1.0.7	fornecer a funcionalidade de auditoria
pgcrypto	1.2	funções criptográficas
pgrouting	2.3.0	Extensão de pgRouting
pgrowlocks	1,1	mostra informações de bloqueio no nível de linha
pgstattuple	1,3	mostra estatísticas no nível da tupla
pg_buffercache	1,1	examina o cache do buffer compartilhado
pg_partman	2.6.3	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1.0	pré-aquecer dados de relações
pg_stat_statements	1,3	acompanhar as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1,1	medição de similaridade de texto e da pesquisa de índice com base em trigrama
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
postgis	2.3.0	Funções e tipos espaciais de geometria, geografia e rasterização do PostGIS
postgis_sfsgal	2.3.0	Funções SFCGAL do PostGIS

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
postgis_tiger_geocoder	2.3.0	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	2.3.0	Tipos e funções espaciais da topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
tablefunc	1.0	funções que manipulam tabelas inteiras, incluindo a tabela de referência cruzada
unaccent	1.0	dicionário de pesquisa de texto que remove sotaques
uuid-ossp	1.0	gera UUIDs (identificadores universais exclusivos)

pg_stat_statements

A extensão [pg_stat_statements](#) é pré-carregada em cada servidor do Banco de Dados do Azure para PostgreSQL para fornecer a você uma forma de acompanhar as estatísticas de execução das instruções SQL. A configuração `pg_stat_statements.track`, que controla quais instruções são contadas por extensão, tem `top` como padrão, que significa que todas as instruções emitidas diretamente por clientes são rastreadas. Dois outros níveis de rastreamento são `none` e `all`. Essa definição é configurável como um parâmetro de servidor por meio de [portal do Azure](#) ou da [CLI do Azure](#).

Há um equilíbrio entre as informações de execução de consulta fornecida por `pg_stat_statements` e o impacto no desempenho do servidor, que registra cada instrução SQL. Se você não está usando ativamente a extensão `pg_stat_statements`, recomendamos que você defina `pg_stat_statements.track` como `none`. Observe que alguns serviços de monitoramento de terceiros podem depender de `pg_stat_statements` para fornecer informações de desempenho de consulta, para confirmar se este é o caso para você ou não.

dblink e postgres_fdw

Com [dblink](#) e [postgres_fdw](#), você pode se conectar de um servidor PostgreSQL a outro ou a outro banco de dados no mesmo servidor. O servidor de recebimento precisa permitir conexões do servidor de envio por meio de seu firewall. Ao usar essas extensões para conectar-se entre os servidores do Banco de Dados do Azure para PostgreSQL, isso pode ser feito definindo "Permitir acesso aos serviços do Azure" como LIGADO. Isso também será necessário se você quiser usar as extensões para executar um loop no mesmo servidor. A configuração "Permitir acesso aos serviços do Azure" pode ser encontrada na página do portal do Azure para o servidor Postgres em Segurança de Conexão. A ativação de "Permitir acesso aos serviços do Azure" coloca todos os IPs do Azure na lista de permitidos.

Atualmente, não há suporte para conexões de saída do Banco de Dados do Azure para PostgreSQL, exceto as conexões com outros servidores do Banco de Dados do Azure para PostgreSQL na mesma região.

uuid

Se você estiver planejando usar `uuid_generate_v4()` na extensão [uuid-ossp](#), considere a possibilidade de comparação com `gen_random_uuid()` na extensão [pgcrypto](#) para obter os benefícios de desempenho.

pgAudit

A extensão pgAudit fornece log de auditoria de sessão e objeto. Para saber como usar essa extensão no Banco de Dados do Azure para PostgreSQL, acesse o [artigo sobre conceitos de auditoria](#).

pg_prewarm

A extensão pg_prewarm carrega dados relacionais no cache. O pré-aquecimento dos caches significa que as consultas têm tempos de resposta melhores na primeira execução após uma reinicialização. No Postgres 10 e inferior, o pré-aquecimento é feito manualmente por meio da [função prewarm](#).

No Postgres 11 e posterior, você pode configurar o pré-aquecimento **automático**. Você precisa incluir pg_prewarm na lista de parâmetros `shared_preload_libraries` e reiniciar o servidor para aplicar a alteração. Os parâmetros podem ser definidos no [portal do Azure](#), na [CLI](#), na API REST ou no modelo do ARM.

TimescaleDB

O TimescaleDB é um banco de dados de série temporal empacotado como uma extensão para PostgreSQL. Ele fornece funções analíticas orientadas a tempo, otimizações e escala o Postgres para cargas de trabalho de série temporal.

Saiba mais sobre o [TimescaleDB](#), uma marca comercial registrada da [Timescale, Inc.](#). O Banco de Dados do Azure para PostgreSQL fornece a [edição Apache-2](#) do TimescaleDB.

Como instalar o TimescaleDB

Para instalar o TimescaleDB, você precisa incluí-lo nas bibliotecas de pré-carregamento compartilhadas do servidor. Uma alteração no parâmetro `shared_preload_libraries` do Postgres exige uma **reinicialização do servidor** para entrar em vigor. Altere os parâmetros usando o [portal do Azure](#) ou a [CLI do Azure](#).

Usando o [portal do Azure](#):

1. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL.
2. Na barra lateral, selecione **Parâmetros do Servidor**.
3. Pesquise o parâmetro `shared_preload_libraries`.
4. Selecione **TimescaleDB**.
5. Escolha **Salvar** para preservar as alterações. Você receberá uma notificação quando a alteração for salva.
6. Após a notificação, **reinic peace** o servidor para aplicar essas alterações. Para saber como reiniciar um servidor, confira [Reiniciar um servidor de Banco de Dados do Azure para PostgreSQL](#).

Agora você pode habilitar TimescaleDB no seu banco de dados do Postgres. Conecte-se ao banco de dados e emita o seguinte comando:

```
CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;
```

TIP

Se você receber um erro, confirme se você [reiniciou o servidor](#) depois de salvar `shared_preload_libraries`.

Agora você pode criar uma hipertabela do TimescaleDB [do zero](#) ou migrar os [dados de série temporal existentes no PostgreSQL](#).

Como restaurar um banco de dados do Timescale

Para restaurar um banco de dados do Timescale usando pg_dump e pg_restore, você precisará executar dois procedimentos auxiliares no banco de dados de destino: `timescaledb_pre_restore()` e `timescaledb_post_restore()`.

Primeiro, prepare o banco de dados de destino:

```
--create the new database where you'll perform the restore
CREATE DATABASE tutorial;
\c tutorial --connect to the database
CREATE EXTENSION timescaledb;

SELECT timescaledb_pre_restore();
```

Agora você pode executar pg_dump no banco de dados original e executar pg_restore. Após a restauração, lembre-se de executar o seguinte comando no banco de dados restaurado:

```
SELECT timescaledb_post_restore();
```

Próximas etapas

Se você não vir uma extensão que gostaria de usar, fale conosco. Vote em solicitações existentes ou crie comentários e solicitações em nosso [fórum de comentários](#).

Limites do Servidor Único do Banco de Dados do Azure para PostgreSQL

21/05/2021 • 3 minutes to read

As seções a seguir descrevem a capacidade e os limites funcionais no serviço de banco de dados. Para saber mais sobre as camadas de recurso (computação, memória e armazenamento), confira o artigo [tipo de preço](#).

Número máximo de conexões

O número máximo de conexões por tipo de preço e vCores é mostrado abaixo. O sistema do Azure exige cinco conexões para monitorar o Banco de Dados do Azure para o servidor PostgreSQL.

TIPO DE PREÇO	VCORE(S)	MÁXIMO DE CONEXÕES	MÁXIMO DE CONEXÕES DE USUÁRIO
Basic	1	55	50
Basic	2	105	100
Uso Geral	2	150	145
Uso Geral	4	250	245
Uso Geral	8	480	475
Uso Geral	16	950	945
Uso Geral	32	1500	1.495
Uso Geral	64	1900	1895
Otimizado para memória	2	300	295
Otimizado para memória	4	500	495
Otimizado para memória	8	960	955
Otimizado para memória	16	1900	1895
Otimizado para memória	32	1987	1982

Quando as conexões excederem o limite, você poderá receber o seguinte erro:

FATAL: já existem muitos clientes

IMPORTANT

Para obter a melhor experiência, é recomendável usar um pool de conexões como o PgBouncer para gerenciar as conexões com eficiência.

Uma conexão PostgreSQL pode ocupar cerca de 10 MB de memória mesmo estando ociosa. Além disso, o estabelecimento de novas conexões leva tempo. A maioria dos aplicativos solicita muitas conexões de curta duração, o que agrava a essa situação. O resultado é um número menor de recursos disponíveis para sua carga de trabalho real, o que leva à redução do desempenho. Um pool de conexões que diminui as conexões ociosas e reutiliza as conexões existentes ajudará a evitar isso. Para saber mais, acesse nossa [postagem no blog](#).

Limitações funcionais

Operações de dimensionamento

- O dimensionamento dinâmico de e para as camadas de preços básicas não tem suporte no momento.
- Atualmente, não há suporte para diminuir o tamanho de armazenamento do servidor.

Upgrade da versão do servidor

- Não há suporte para a migração automatizada entre versões de mecanismo de banco de dados principal. Se você quiser atualizar para a próxima versão principal, faça um [despejo e restaure](#) para um servidor que foi criado com a nova versão do mecanismo.

Observe que, antes do PostgreSQL versão 10, a [política de versão do PostgreSQL](#) considerou uma atualização de *versão principal* como um aumento no primeiro *ou* segundo número (por exemplo, 9.5 para 9.6 era considerado uma atualização de versão *principal*). A partir da versão 10, apenas uma alteração no primeiro número é considerada uma atualização de versão principal (por exemplo, 10.0 para 10.1 é uma atualização de versão *secundária* e 10 a 11 é uma atualização de versão *principal*).

Ponto de extremidade de serviço VNet

- O suporte para ponto de extremidade de serviço de VNet é apenas para servidores de Uso Geral e Otimizados para Memória.

Restaurando um servidor

- Ao usar o recurso PITR, o novo servidor é criado com as mesmas configurações de camadas de preços que o servidor em que é baseado.
- O novo servidor criado durante uma restauração não possui as regras de firewall existentes no servidor original. As regras de firewall precisam ser configuradas separadamente para esse novo servidor.
- Não há suporte para restaurar um servidor eliminado.

Caracteres UTF-8 no Windows

- Em alguns cenários, não há suporte completo para caracteres UTF-8 no PostgreSQL de software livre no Windows, o que afeta o Banco de Dados do Azure para PostgreSQL. Confira o thread [Bug nº 15476 nos arquivos do PostgreSQL](#) para obter mais informações.

Erro GSS

Se aparecer um erro relacionado a GSS, provavelmente você está usando uma versão de cliente/driver mais recente que o servidor único do Azure Postgres ainda não oferece compatibilidade total. Esse erro é conhecido por afetar [versões 42.2.15 e 42.2.16 do driver JDBC](#).

- Planejamos concluir a atualização até o final de novembro. Considere usar uma versão de driver que esteja funcionando enquanto isso.
- Ou considere desabilitar a solicitação de GSS. Use um parâmetro de conexão como `gssEncMode=disable`.

Redução do tamanho do armazenamento

O tamanho do armazenamento não pode ser reduzido. É necessário criar um servidor com o tamanho de armazenamento desejado, executar o [despejar e restaurar](#) manualmente e migrar os bancos de dados para o novo servidor.

Próximas etapas

- Entenda [o que está disponível em cada tipo de preço](#)
- Saiba mais sobre [Versões de Banco de Dados PostgreSQL com suporte](#)
- Veja [Como fazer backup e restaurar um servidor no Banco de Dados do Azure para PostgreSQL usando o Portal do Azure](#)

Tipos de preço no Banco de Dados do Azure para PostgreSQL - Servidor único

25/05/2021 • 8 minutes to read

É possível criar um servidor do Banco de Dados do Azure para PostgreSQL em um dos três tipos de preço diferentes: Básico, Uso Geral e Otimizado para Memória. Os tipos de preço são diferenciados pela quantidade de computação nos vCores que pode ser provisionada, pela memória por vCore e pela tecnologia de armazenamento usada para armazenar os dados. Todos os recursos são provisionados no nível do servidor PostgreSQL. Um servidor pode ter um ou vários bancos de dados.

RECURSO/CAMADA	BASIC	USO GERAL	OTIMIZADO PARA MEMÓRIA
Geração de computação	Gen 4, Gen 5	Gen 4, Gen 5	Gen 5
vCores	1, 2	2, 4, 8, 16, 32, 64	2, 4, 8, 16, 32
Memória por vCore	2 GB	5 GB	10 GB
Tamanho de armazenamento	5 GB a 1 TB	5 GB a 16 TB	5 GB a 16 TB
Período de retenção do backup de banco de dados	7 a 35 dias	7 a 35 dias	7 a 35 dias

Para escolher um tipo de preço, use a tabela a seguir como ponto de partida.

TIPO DE PREÇO	CARGAS DE TRABALHO DE DESTINO
Básico	Cargas de trabalho que exigem desempenho de E/S e computação leve. Os exemplos incluem servidores usados para desenvolvimento ou teste ou aplicativos de pequena escala usados com pouca frequência.
Uso Geral	A maioria das cargas de trabalho que exigem a computação e a memória balanceadas com a taxa de transferência de E/S escalonável. Os exemplos incluem servidores para hospedar aplicativos Web e móveis e outros aplicativos empresariais.
Otimizado para memória	Cargas de trabalho de banco de dados de alto desempenho que exigem desempenho na memória para o processamento de transações mais rápido e com simultaneidade mais alta. Os exemplos incluem servidores para o processamento de dados em tempo real e aplicativos analíticos ou transacionais de alto desempenho.

Depois de criar um servidor, o número de vCores a geração de hardware e o tipo de preço (exceto em Básico) pode ser alterado para cima ou para baixo em segundos. Você pode também, independentemente, ajustar a quantidade de armazenamento de backup e o período de retenção de backup para cima ou para baixo sem tempo de inatividade do aplicativo. Não será possível alterar o tipo de armazenamento de backup depois que um servidor é criado. Para obter mais informações, consulte a seção [Recursos de dimensionamento](#).

Gerações de computação e vCores

Os recursos de computação são fornecidos como vCores, que representam a CPU lógica do hardware subjacente. Leste da China 1, Norte da China 1, US DoD Central e Leste do US DoD usam CPUs lógicas de geração 4 baseadas em processadores Intel E5-2673 v3 (Haswell) de 2,4 GHz. Todas as outras regiões usam CPUs lógicas de geração 5 baseadas em processadores Intel E5-2673 v4 (Broadwell) de 2,3 GHz.

Armazenamento

O armazenamento provisionado é a quantidade de capacidade de armazenamento disponível para o Banco de Dados do Azure para servidor PostgreSQL. O armazenamento é usado para os arquivos de banco de dados, os logs de transações e os logs do servidor PostgreSQL. A quantidade total de armazenamento que você provisão também define a capacidade disponível para o servidor.

ATRIBUTOS DE ARMAZENAMENTO	BASIC	USO GERAL	OTIMIZADO PARA MEMÓRIA
Tipo de armazenamento	Armazenamento Básico	Armazenamento de Uso Geral	Armazenamento de Uso Geral
Tamanho de armazenamento	5 GB a 1 TB	5 GB a 16 TB	5 GB a 16 TB
Tamanho do incremento de armazenamento	1 GB	1 GB	1 GB
IOPS	Variável	3 IOPS/GB Mín 100 IOPS Máx. 20.000 IOPS	3 IOPS/GB Mín 100 IOPS Máx. 20.000 IOPS

NOTE

Há suporte para o armazenamento de até 16 TB e 20.000 IOPS nas seguintes regiões: Leste da Austrália, Sudeste da Austrália, Sul do Brasil, Canadá Central, Leste do Canadá, EUA Central, Leste da China 2, Norte da China 2, Leste da Ásia, Leste dos EUA, Leste dos EUA 1, Leste dos EUA 2, Leste do Japão, Oeste do Japão, Coreia Central, Sul da Coreia, Centro-Norte dos EUA, Norte da Europa, Centro-Sul dos EUA, Sudeste da Ásia, Norte da Suíça, Oeste da Suíça, Leste do US Gov, Centro-Sul do US Gov, Sudoeste do US Gov, Sul do Reino Unido, Oeste do Reino Unido, Oeste da Europa, Centro-Oeste dos EUA, Oeste dos EUA e Oeste dos EUA 2.

Todas as outras regiões dão suporte a até 4 TB de armazenamento e 6.000 IOPS.

Você pode adicionar mais capacidade de armazenamento durante e após a criação do servidor e permitir que o sistema aumente o armazenamento automaticamente com base no consumo de armazenamento de sua carga de trabalho.

NOTE

O armazenamento só pode ser escalado verticalmente, não horizontalmente.

A camada Básico não oferece garantia de IOPS. Nos tipos de preço Uso Geral e Otimizado para Memória, o IOPS é dimensionado com o tamanho de armazenamento provisionado a uma taxa de 3:1.

Você pode monitorar o consumo de E/S no Portal do Azure ou usando os comandos da CLI do Azure. As métricas relevantes para monitorar são o [limite de armazenamento](#), [porcentagem de armazenamento](#), [armazenamento usado](#) e [porcentagem de E/S](#).

Alcançando o limite de armazenamento

Os servidores com armazenamento provisionado menor ou igual a 100 GB serão marcados como somente leitura caso o armazenamento livre seja inferior a 512 MB ou 5% do tamanho do armazenamento provisionado. Os servidores com mais de 100 GB de armazenamento provisionado serão marcados como somente leitura quando o armazenamento livre for inferior a 5 GB.

Por exemplo, se você provisionou 110 GB de armazenamento e a utilização real superar 105 GB, o servidor será marcado como somente leitura. Como alternativa, se você tiver provisionado 5 GB de armazenamento, o servidor será somente leitura quando o armazenamento livre atingir menos de 512 MB.

Quando o servidor é definido como somente leitura, todas as sessões existentes são desconectadas e as transações não confirmadas são revertidas. Todas as operações de gravação subsequente e a transação falham. Todas as consultas de leitura continuam a funcionar sem interrupções.

Você pode aumentar a quantidade de armazenamento provisionado para o servidor ou iniciar uma nova sessão no modo de gravação de leitura e soltar os dados para recuperar o armazenamento livre. Executar

```
SET SESSION CHARACTERISTICS AS TRANSACTION READ WRITE;
```

define a sessão atual para o modo de gravação de leitura. Para evitar a corrupção de dados, não execute nenhuma operação de gravação quando o servidor ainda estiver em status somente leitura.

Recomendamos que você ative o aumento automático do armazenamento ou configure um alerta para notificá-lo quando o armazenamento do servidor estiver se aproximando do limite, para evitar entrar no estado somente leitura. Para mais informações, consulte a documentação em [como configurar um alerta](#).

Aumento automático do armazenamento

O aumento automático do armazenamento impede que o servidor fique sem armazenamento e se torne somente leitura. Se o aumento automático do armazenamento estiver habilitado, o armazenamento aumentará automaticamente sem afetar a carga de trabalho. Para servidores com armazenamento provisionado menor ou igual a 100 GB, o tamanho do armazenamento provisionado será aumentado em 5 GB assim que o armazenamento gratuito estiver abaixo de 1 GB ou 10% do armazenamento provisionado. Para servidores com mais de 100 GB de armazenamento provisionado, o tamanho do armazenamento provisionado aumenta em 5% quando o espaço livre de armazenamento está abaixo de 10 GB ou 5% do tamanho de armazenamento provisionado, o que for maior. Os limites máximos de armazenamento conforme especificados anteriormente se aplicam.

Por exemplo, se você provisionou 1.000 GB de armazenamento e a utilização real passar de 950 GB, o tamanho do armazenamento será aumentado para 1.050 GB. Como alternativa, se você tiver provisionado 10 GB de armazenamento, o tamanho do armazenamento aumentará para 15 GB quando menos de 1 GB de armazenamento for gratuito.

Não esqueça de que o armazenamento só pode ser escalado verticalmente, não horizontalmente.

Armazenamento de backup

O Banco de Dados do Azure para PostgreSQL fornece até 100% de seu armazenamento de servidor configurado como armazenamento de backup sem custo adicional. Qualquer armazenamento de backup usado além desse valor é cobrado em GB por mês. Por exemplo, se você provisionar um servidor com 250 GB de armazenamento, terá 250 GB de armazenamento adicional disponível para backups de servidor, sem custo adicional. O armazenamento para backups além de 250 GB será cobrado de acordo com o [modelo de preços](#). Para entender os fatores que influenciam o uso do armazenamento de backup, o monitoramento e o controle do custo de armazenamento de backup, consulte [documentação de backup](#).

Escalar recursos

Após criar o servidor, você poderá, independentemente, alterar vCores, a geração de hardware, o tipo de preço

(exceto em Básico), a quantidade de armazenamento e o período de retenção de backup. Não será possível alterar o tipo de armazenamento de backup depois que um servidor é criado. O número de vCores pode ser dimensionado para cima ou para baixo. Os vCores e o período de retenção de backup podem ser aumentados ou reduzidos de 7 a 35 dias. O tamanho de armazenamento só pode ser aumentado. O dimensionamento dos recursos pode ser feito por meio do portal ou da CLI do Azure. Para obter um exemplo de dimensionamento usando a CLI do Azure, consulte [Monitorar e dimensionar um servidor do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#).

NOTE

O tamanho de armazenamento só pode ser aumentado. Você não poderá voltar para um tamanho de armazenamento menor após o aumento.

Ao alterar o número de vCores, a geração de hardware ou o tipo de preço, uma cópia do servidor original é criada com a nova alocação de computação. Depois que o novo servidor entra em execução, as conexões são alternadas para o novo servidor. Durante um momento enquanto o sistema muda para o novo servidor, nenhuma nova conexão pode ser estabelecida e todas as transações não confirmadas são revertidas. Esse período varia, mas na maioria dos casos fica abaixo um minuto.

O dimensionamento do armazenamento e a alteração do período de retenção de backup são operações realmente online. Não há nenhum tempo de inatividade e o aplicativo não é afetado. Conforme o IOPS é dimensionado com o tamanho do armazenamento provisionado, você pode aumentar o IOPS disponível para seu servidor aumentando o armazenamento.

Preços

Para as informações mais recentes sobre preços, consulte a [página de preços do serviço](#). Para ver os custos da configuração desejada, o [Portal do Azure](#) mostra o custo mensal na guia **Tipo de preço** com base nas opções que você seleciona. Se você não tiver uma assinatura do Azure, poderá usar a calculadora de preços do Azure para obter um preço estimado. No site da [Calculadora de preços do Azure](#), selecione **Adicionar itens**, expanda a categoria **Bancos de dados** e escolha **Banco de Dados do Azure para PostgreSQL** para personalizar as opções.

Próximas etapas

- Saiba como [criar um servidor PostgreSQL no portal](#).
- Conheça os [limites de serviço](#).
- Saiba como [fazer a expansão com réplicas de leitura](#).

Pagar antecipadamente pelos recursos de computação de servidor único do Banco de Dados do Azure para PostgreSQL com capacidade reservada

09/08/2021 • 7 minutes to read

O Banco de Dados do Azure para PostgreSQL agora ajuda você a economizar, com o pagamento antecipado de recursos de computação em comparação com o preço pago conforme o uso. Com a capacidade reservada do Banco de Dados do Azure para PostgreSQL, você firma um compromisso antecipado no servidor PostgreSQL por um período de um ou três anos para receber um desconto considerável nos custos de computação. Para comprar a capacidade reservada do Banco de Dados do Azure para PostgreSQL, é necessário especificar a região do Azure, o tipo de implantação, o nível de desempenho e o prazo.

Você não precisa atribuir a reserva a servidores específicos do Banco de Dados do Azure para PostgreSQL. Um Banco de Dados do Azure para PostgreSQL já em execução (ou aqueles que foram implantados recentemente) obterá automaticamente o benefício do preço reservado. Ao comprar uma reserva, você está pagando antecipadamente os custos de computação por um período de um ou três anos. Assim que você compra uma reserva, as cobranças de processamento do Banco de Dados do Azure para PostgreSQL que correspondem aos atributos de reserva não são mais feitas como «pague conforme usar». Uma reserva não cobre custos de software, rede ou armazenamento associados à instância dos servidores de Banco de Dados PostgreSQL. No final do período de reserva, o benefício de cobrança expira, e o Banco de Dados do Azure para PostgreSQL é cobrado pelo preço de pré-pagamento. As reservas não são renovadas automaticamente. Para obter informações sobre preços, consulte a [oferta de capacidade reservada do Banco de Dados do Azure para PostgreSQL](#).

IMPORTANT

O preço da capacidade reservada está disponível para o Banco de Dados do Azure para PostgreSQL nas opções de implantação de Servidor único e de [Citus de hiperescala](#). Para obter informações sobre os preços de RI em Hiperescala (Citus), consulte [esta página](#).

Você pode comprar capacidade reservada do Banco de Dados do Azure para PostgreSQL no [portal do Azure](#). Pague pela reserva [antecipadamente ou com pagamentos mensais](#). Para comprar a capacidade reservada:

- Você deve ter a função de Proprietário em pelo menos uma assinatura corporativa ou individual com tarifas de pagamento conforme o uso.
- Para as assinaturas Enterprise, a opção **Adicionar Instâncias Reservadas** deve estar habilitada no [Portal EA](#). Ou, se essa configuração estiver desabilitada, você deve ser um administrador de EA na assinatura.
- Para o programa Provedor de Solução de Nuvem (CSP), apenas os agentes admin ou agentes de vendas podem adquirir a capacidade reservada do Banco de Dados do Azure para PostgreSQL.

Para saber detalhes sobre como clientes empresariais e clientes do método de pagamento conforme o uso são cobrados para compras de reserva, confira [Entenda o uso de reserva do Azure para seu registro Enterprise](#) e [Entenda o uso de reserva do Azure para sua assinatura paga conforme o uso](#).

Determinar o tamanho correto do servidor antes da compra

O tamanho da reserva deve ser baseado na quantidade total de processamento usada pelos servidores existentes ou prestes a serem implantados dentro de uma região específica, e usando o mesmo nível de desempenho e geração de hardware.

Por exemplo, vamos supor que você esteja executando um banco de dados PostgreSQL vCore Gen5 - 32 de uso geral e dois bancos de dados PostgreSQL vCore Gen5 - 16 com otimização de memória. Além disso, vamos supor que você planeje implantar, no próximo mês, um servidor de banco de dados vCore Gen5 – 8 de uso geral e um servidor de banco de dados vCore Gen5 – 32 com otimização de memória. Vamos supor que você saiba que precisará desses recursos por pelo menos um ano. Nesse caso, você deve comprar um vCore 40 (32 + 8), reserva de um ano para o único banco de dados de uso geral - Gen5 e um vCore 64 (2x16 + 32) de reserva de um ano para o banco de dados único otimizado para memória - Gen5

Comprar capacidade reservada do Banco de Dados do Azure para PostgreSQL

1. Entre no [portal do Azure](#).
2. Selecione **Todos os serviços > Reservas**.
3. Selecione **Adicionar** e, em seguida, no painel reservas de compra, selecione **Banco de Dados do Azure para PostgreSQL** para comprar uma nova reserva para seus bancos de dados PostgreSQL.
4. Preencha os campos obrigatórios. Bancos de dados novos ou existentes que correspondem aos atributos que você selecionou se qualificam para obter o desconto de capacidade reservada. O número real de servidores de Banco de Dados do Azure para PostgreSQL que recebem o desconto depende do escopo e da quantidade selecionada.

Select the product you want to purchase X

Save on your Azure Database for PostgreSQL single server reserved vCores costs by purchasing reserved capacity. Discount will apply automatically to the matching database deployments. The discount applies hourly on the compute usage, unused reserved capacity doesn't carry over. Discount only applies to single server deployment. [Learn More](#)

Scope * (Single subscription) Subscription * (Pricing Webapp)

Filter by name... Region : **East US** Billing frequency : **Select a value** (X) Reset filters

Name	Region	Term	Billing frequency
Azure Database for PostgreSQL General Purpose - Compute Gen5	East US	One Year	Upfront
Azure Database for PostgreSQL General Purpose - Compute Gen5	East US	One Year	Monthly
Azure Database for PostgreSQL Memory Optimized - Compute Gen5	East US	One Year	Upfront
Azure Database for PostgreSQL Memory Optimized - Compute Gen5	East US	One Year	Monthly

Select Cancel

A tabela a seguir descreve os campos obrigatórios.

CAMPO	DESCRIÇÃO
-------	-----------

CAMPO	DESCRIÇÃO
Subscription	A assinatura usada para a reserva de capacidade reservada do Banco de Dados do Azure para PostgreSQL. O método de pagamento na assinatura é cobrado pelos custos iniciais da reserva de capacidade reservada do Banco de Dados do Azure para PostgreSQL. O tipo de assinatura deve ser um contrato corporativo (números de oferta MS-AZR-0017P ou MS-AZR-0148P) ou um contrato individual de pagamento conforme o uso (números de oferta MS-AZR-0003P ou MS-AZR-0023P). Para uma assinatura corporativa, os encargos são deduzidos do saldo do Pagamento antecipado do Azure do registro (chamado anteriormente de compromisso monetário) ou cobrados como excedente. Para uma assinatura individual com taxas pagas conforme o uso, as cobranças são feitas na forma de pagamento de cartão de crédito ou de fatura na assinatura.
Escopo	<p>O escopo da reserva vCore pode cobrir uma assinatura ou várias assinaturas (escopo compartilhado). Se você selecionar:</p> <p>Compartilhado, o desconto de reserva de vCore será aplicado aos servidores de Banco de Dados do Azure para PostgreSQL em execução em todas as assinaturas dentro de seu contexto de cobrança. Para clientes empresariais, o escopo compartilhado é o registro e inclui todas as assinaturas no registro. Para clientes de Pagamento Conforme o Uso, o escopo compartilhado consiste em todas as assinaturas de Pagamento Conforme o Uso criadas pelo administrador da conta.</p> <p>Assinatura única, o desconto de reserva do vCore é aplicado aos servidores de Banco de Dados do Azure para PostgreSQL nesta assinatura.</p> <p>Se você selecionar Um único grupo de recursos, o desconto de reserva será aplicado aos servidores de Banco de Dados do Azure para PostgreSQL na assinatura selecionada e ao grupo de recursos selecionado nessa assinatura.</p>
Região	A região do Azure abrangida pela reserva de capacidade reservada do Banco de Dados do Azure para PostgreSQL.
Tipo de implantação	O tipo de recurso do Banco de Dados do Azure para PostgreSQL para o qual você deseja comprar a reserva.
Nível de desempenho	A camada de serviço dos servidores de Banco de Dados PostgreSQL do Azure.
Termo	Um ano

CAMPO	DESCRIÇÃO
Quantidade	A quantidade de recursos de computação que estão sendo comprados na reserva de capacidade reservada do Banco de Dados PostgreSQL do Azure. A quantidade é um número de vCores na região do Azure selecionada e o nível de desempenho que está sendo reservado e receberão o desconto de cobrança. Por exemplo, se você estiver executando ou planejando executar servidores de Banco de Dados PostgreSQL do Azure com a capacidade de computação total de 16 vCores Gen5 na região Leste dos EUA, você deverá especificar a quantidade como 16 para maximizar o benefício para todos os servidores.

Cancelar, trocar ou reembolsar reservas

É possível cancelar, trocar ou reembolsar reservas com determinadas limitações. Para saber mais, confira [Trocas e reembolsos via autoatendimento para Reservas do Azure](#).

Flexibilidade de tamanho do vCore

A flexibilidade de tamanho do vCore ajuda você a aumentar ou diminuir dentro de uma região e nível de desempenho sem perder o benefício de capacidade reservada. Se você dimensionar para mais vCores maiores do que a capacidade reservada, haverá cobrança pelo excesso de vCores usando o preço pago conforme o uso.

Precisa de ajuda? Fale conosco

Se você tiver dúvidas ou precisar de ajuda, [crie uma solicitação de suporte](#).

Próximas etapas

O desconto de reserva de vCore é aplicado automaticamente ao número de servidores de Banco de Dados PostgreSQL do Azure que correspondem aos atributos e ao escopo de reserva de capacidade reservada do Banco de Dados PostgreSQL do Azure. Você pode atualizar o escopo da reserva de capacidade reservada do Banco de Dados PostgreSQL do Azure por meio do portal do Azure, PowerShell, CLI ou por meio da API.

Para saber mais sobre as Reservas do Azure, consulte os seguintes artigos:

- [O que são Reservas do Azure?](#)
- [Gerenciar Reservas do Azure](#)
- [Compreender o desconto de Reservas do Azure](#)
- [Entender o uso de reserva para a sua assinatura paga conforme o uso](#)
- [Entender o uso de reserva para seu registro de empresa](#)
- [Reservas do Azure no programa de CSP \(Provedor de Soluções na Nuvem\) do Partner Center](#)

Segurança no Banco de Dados do Azure para PostgreSQL – Servidor Único

01/07/2021 • 3 minutes to read

Há várias camadas de segurança disponíveis para proteger os dados em seu servidor do Banco de Dados do Azure para PostgreSQL. Este artigo descreve essas opções de segurança.

Proteção e criptografia de informações

Em trânsito

O Banco de Dados do Azure para PostgreSQL protege seus dados criptografando os dados em trânsito com o protocolo TLS. A criptografia (SSL/TLS) é imposta por padrão.

Em repouso

O serviço Banco de Dados do Azure para PostgreSQL usa o módulo de criptografia validado por FIPS 140-2 para criptografia de armazenamento de dados em repouso. Os dados, incluindo backups, são criptografados no disco, assim como os arquivos temporários criados durante a execução de consultas. O serviço usa a criptografia AES de 256 bits incluída na criptografia de armazenamento do Azure e as chaves são gerenciadas pelo sistema. A criptografia de armazenamento está sempre ativada e não pode ser desabilitada.

Segurança de rede

As conexões com um servidor de Banco de Dados do Azure para PostgreSQL são encaminhadas primeiro por um gateway regional. O gateway tem um IP acessível publicamente, enquanto os endereços IP do servidor são protegidos. Para saber mais sobre o gateway, confira o [artigo sobre arquitetura de conectividade](#).

Um servidor de Banco de Dados do Azure para PostgreSQL criado recentemente tem um firewall que bloqueia todas as conexões externas. Embora atinjam o gateway, elas não têm permissão para se conectar ao servidor.

Regras de firewall de IP

As regras de firewall de IP permitem acesso ao servidor com base no endereço IP de origem de cada solicitação. Confira a [visão geral das regras de firewall](#) para obter mais informações.

Regras de firewall de rede virtual

Os pontos de extremidade de serviço de rede virtual estendem sua conectividade de rede virtual no backbone do Azure. Usando regras de rede virtual, você pode habilitar seu Banco de Dados do Azure para PostgreSQL para permitir conexões de sub-redes selecionadas em uma rede virtual. Para saber mais, confira a [visão geral do ponto de extremidade de serviço de rede virtual](#).

IP Privado

O Link Privado permite que você se conecte ao seu Banco de Dados do Azure para PostgreSQL (Servidor único) no Azure por meio de um ponto de extremidade privado. O Link Privado do Azure essencialmente traz os serviços do Azure dentro de sua VNet (Rede Virtual privada). Os recursos de PaaS podem ser acessados usando o endereço IP privado, assim como qualquer outro recurso na VNet. Para saber mais, confira [visão geral do link privado](#).

Gerenciamento de acesso

Ao criar o servidor do Banco de Dados do Azure para PostgreSQL, você fornece credenciais para uma função de

administrador. Tal administrador pode ser usado para criar [funções adicionais do PostgreSQL](#).

Você pode se conectar ao servidor usando a [autenticação do AAD \(Azure Active Directory\)](#).

Proteção contra ameaças

Você pode aceitar a [Proteção Avançada contra Ameaças](#) que detecta atividades anômalas, indicando tentativas incomuns e potencialmente prejudiciais de acessar ou explorar os servidores.

O [log de auditoria](#) está disponível para acompanhar a atividade em seus bancos de dados.

Migrar da Oracle

O Oracle dá suporte a TDE (Transparent Data Encryption) para criptografar dados de tabela e de espaço de tabela. No Azure para PostgreSQL, os dados são criptografados automaticamente em várias camadas. Confira a seção "Em repouso" nesta página e também vários tópicos de segurança, incluindo [Chaves gerenciadas pelo cliente](#) e [Criptografia dupla de infraestrutura](#). Você também pode considerar o uso da extensão [pgcrypto](#), que tem suporte no [Azure para PostgreSQL](#).

Próximas etapas

- Habilitar regras de firewall para [IPS](#) ou [redes virtuais](#)
- Saiba mais sobre a [autenticação do Active Directory](#) no Banco de Dados do Azure para PostgreSQL

Configurar a conectividade do TLS no Servidor Único do Banco de Dados do Azure para PostgreSQL

16/06/2021 • 5 minutes to read

O Banco de Dados do Azure para PostgreSQL prefere conectar os aplicativos cliente ao serviço do PostgreSQL que usam o protocolo TLS, anteriormente conhecido como protocolo SSL. Impor conexões TLS entre seu servidor de banco de dados e os aplicativos clientes ajuda a proteger contra ataques de "intermediários" ao criptografar o fluxo de dados entre o servidor e seu aplicativo.

Por padrão, o serviço de banco de dados do PostgreSQL é configurado para exigir conexão do TLS. Você pode optar por desabilitar a necessidade de TLS se o aplicativo cliente não oferecer suporte à conectividade TLS.

NOTE

Com base nos comentários dos clientes, estendemos a substituição do certificado raiz para nossa CA Raiz Baltimore existente até 15 de fevereiro de 2021 (15/02/2021).

IMPORTANT

O certificado raiz SSL está definido para expirar a partir de 15 de fevereiro de 2021 (15/02/2021). Atualize o aplicativo para usar o [novo certificado](#). Para saber mais, confira [atualizações de certificado planejadas](#)

Impondo conexões do protocolo TLS

Para todos os Bancos de Dados do Azure para servidores PostgreSQL provisionados com o Portal e a CLI do Azure, a imposição de conexões TSL está habilitada por padrão.

Da mesma forma, cadeias de conexão previamente definidas nas configurações de "Cadeias de Conexão" em seu servidor no Portal do Azure incluem os parâmetros necessários para linguagens comuns a fim de se conectar ao seu servidor de banco de dados usando TSL. O parâmetro TSL varia de acordo com o conector, por exemplo "ssl=true" ou "sslmode=require" ou "sslmode=required" e outras variações.

Configurar a imposição do protocolo TSL

Como opção, você pode desabilitar a imposição da conectividade TSL. O Microsoft Azure recomenda sempre habilitar a configuração **Impor conexão SSL** para melhorar a segurança.

Usando o portal do Azure

Visite seu servidor de Banco de Dados do Azure para PostgreSQL e clique em **Segurança de conexão**. Use o botão de alternância para habilitar ou desabilitar a configuração **Impor conexão SSL**. Em seguida, clique em **Salvar**.

The screenshot shows the 'myserver-20170401 - Connection security' page in the Azure portal. On the left, a sidebar lists 'SETTINGS' and 'Connection security' (which is highlighted with a red box). The main area shows 'SSL settings' with a note about enforcing SSL connections. A button labeled 'Save' is highlighted with a red box. Below it, there's a toggle switch between 'Enabled' and 'Disabled', which is set to 'Enabled'. Under 'Firewall rules', a table shows a single rule named 'AllowAllips' with 'START IP' as '0.0.0.0' and 'END IP' as '255.255.255.255'.

Você pode confirmar a configuração exibindo a página **Visão geral** para ver o indicador **Status de imposição de SSL**.

Usando a CLI do Azure

Você pode habilitar ou desabilitar o parâmetro `ssl-enforcement` usando os valores `Enabled` ou `Disabled` respectivamente na CLI do Azure.

```
az postgres server update --resource-group myresourcegroup --name mydemoserver --ssl-enforcement Enabled
```

Verificar se o seu aplicativo ou sua estrutura oferece suporte a conexões TSL

Algumas estruturas do aplicativos que usam o PostgreSQL para serviços de banco de dados não habilitam o protocolo TSL por padrão durante a instalação. Se o seu servidor PostgreSQL impõe conexões TLS, mas o aplicativo não está configurado para o TLS, o aplicativo pode falhar ao se conectar ao seu servidor de banco de dados. Confira a documentação de seu aplicativo para saber como habilitar conexões TSL.

Aplicativos que exigem a verificação de certificado para conectividade TSL

Em alguns casos, os aplicativos exigem um arquivo de certificado local gerado de um arquivo de certificado de uma Autoridade de Certificação (CA) confiável para se conectar com segurança. O certificado para se conectar a um servidor de Banco de Dados do Azure para PostgreSQL está localizado em

<https://www.digicert.com/CACerts/BaltimoreCyberTrustRoot.crt.pem>. Baixe o arquivo de certificado e salve-o em seu local preferido.

Consulte os links a seguir para obter certificados para servidores em nuvens soberanas: [Azure Governamental](#), [Azure China](#) e [Azure Alemanha](#).

Conecte-se usando psql

O exemplo a seguir mostra como conectar-se ao servidor PostgreSQL usando o utilitário de linha de comando `psql`. Use a configuração de cadeia de conexão `sslmode=verify-full` para impor a verificação de certificado TLS/SSL. Passe o caminho do arquivo de certificado local para o parâmetro `sslrootcert`.

O comando a seguir é um exemplo da cadeia de conexão `psql`:

```
psql "sslmode=verify-full sslrootcert=BaltimoreCyberTrustRoot.crt host=mydemoserver.postgres.database.azure.com dbname=postgres user=myuser@mydemoserver"
```

TIP

Confirme se o valor passado para `sslrootcert` corresponde ao caminho do arquivo para o certificado que você salvou.

Imposição de TLS no Servidor único do Banco de Dados do Azure para PostgreSQL

O Servidor Único do Banco de dados do Azure para PostgreSQL dá suporte à criptografia para clientes que se conectam ao seu servidor de banco de dados usando o protocolo TLS. O TLS é um protocolo padrão do setor que garante conexões de rede seguras entre o servidor de banco de dados e os aplicativos cliente, permitindo que você atenda aos requisitos de conformidade.

Configurações de protocolo TLS

O Servidor Único do Banco de Dados do Azure para PostgreSQL fornece a capacidade de impor a versão de TLS para as conexões de cliente. Para impor a versão de TLS, use a configuração de opção **versão mínima do TLS**. Os valores a seguir são permitidos para essa configuração de opção:

CONFIGURAÇÃO DE TLS MÍNIMA	VERSÃO DO TLS DO CLIENTE COM SUPORTE
TLSEnforcementDisabled (padrão)	Não é necessário TLS
TLS1_0	TLS 1.0, TLS 1.1, TLS 1.2 e superior
TLS1_1	TLS 1.1, TLS 1.2 e superior
TLS1_2	Versão TLS 1.2 e superior

Por exemplo, definir a versão Mínima de configuração do TLS como 1.0 significa que o servidor permitirá conexões de clientes usando TLS 1.0, 1.1 e 1.2+. Como alternativa, definir como 1.2 significa que você só permitirá conexões de clientes que usam o TLS 1.2+ e todas as conexões com o TLS 1.0 e TLS 1.1 serão rejeitadas.

NOTE

Por padrão, o Banco de Dados do Azure para PostgreSQL não impõe uma versão mínima do TLS (a configuração `TLSEnforcementDisabled`).

Depois de aplicar uma versão mínima do TLS, você não poderá desabilitar mais tarde a imposição mínima da versão.

Para saber como definir a configuração de TLS para o Servidor único do Banco de Dados do Azure para PostgreSQL, consulte [Como definir a configuração de TLS](#).

Suporte à criptografia do Servidor Único do Banco de Dados do Azure para PostgreSQL

Como parte da comunicação do protocolo SSL/TLS, os conjuntos de criptografia são validados e apenas os conjuntos de criptografia com suporte têm permissão para se comunicar com o servidor do banco de dados. A validação do conjunto de criptografia é controlada na [camada do gateway](#) e não explicitamente no próprio nó. Se os conjuntos de criptografia não corresponderem a um dos conjuntos listados abaixo, as conexões de entrada do cliente serão rejeitadas.

Conjunto de criptografia com suporte

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

Próximas etapas

Revise várias opções de conectividade do aplicativo em [Bibliotecas de conexão para o Banco de Dados do Azure para PostgreSQL](#).

- Saiba mais sobre [configuração de TLS](#)

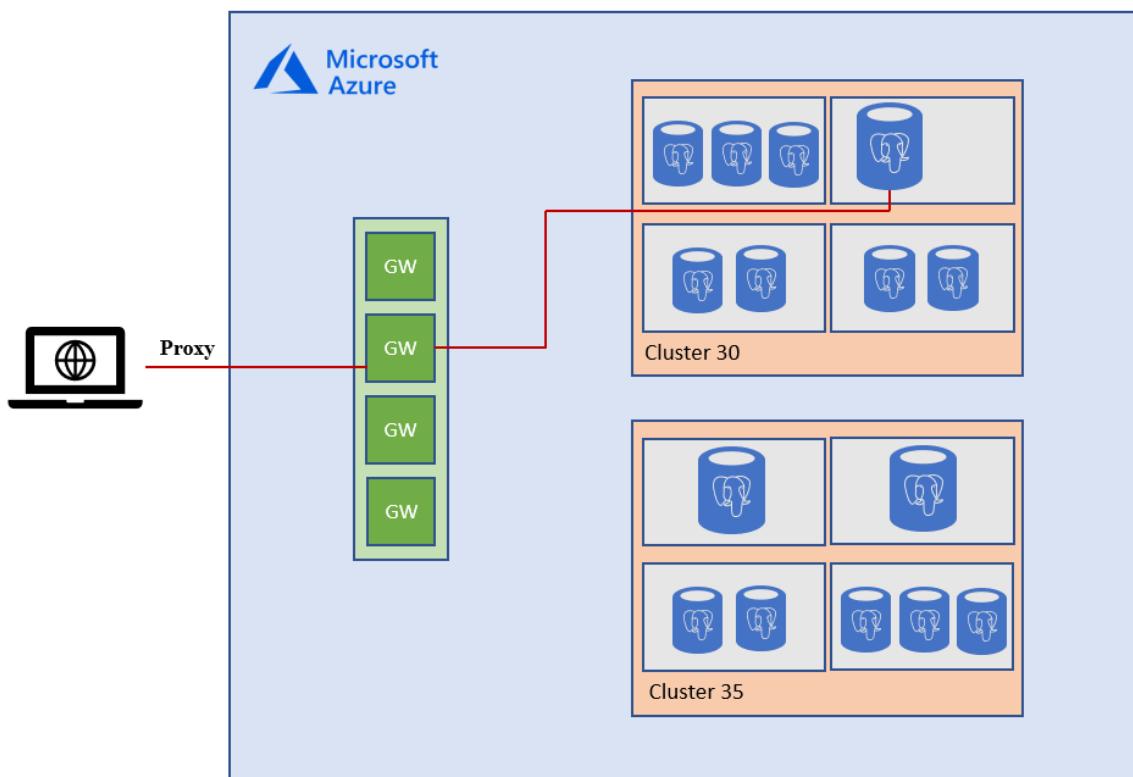
Arquitetura de conectividade no Banco de Dados do Azure para PostgreSQL

21/05/2021 • 8 minutes to read

Este artigo explica a arquitetura de conectividade do Banco de Dados do Azure para PostgreSQL e também como o tráfego é direcionado para a instância do Banco de Dados do Azure para PostgreSQL de clientes dentro e fora do Azure.

Arquitetura de conectividade

A conexão com o Banco de Dados do Azure para PostgreSQL é estabelecida por meio de um gateway responsável por rotear as conexões de entrada até o local físico de seu servidor em nossos clusters. O seguinte diagrama ilustra esse fluxo de tráfego.



À medida que o cliente conecta-se ao banco de dados, a cadeia de conexão do servidor é resolvida para o endereço IP do gateway. O gateway, por padrão, escuta no endereço IP na porta 5432. No cluster do banco de dados, o tráfego é encaminhado para ao Banco de Dados do Azure para PostgreSQL apropriado. Portanto, a fim de conectar-se ao servidor, por exemplo, de redes corporativas, é necessário abrir o **firewall do lado do cliente para permitir que o tráfego de saída seja capaz de acessar nossos gateways**. Abaixo, você pode localizar uma lista completa dos endereços IP usados por nossos gateways por região.

Endereços IP de gateway do Banco de Dados do Azure para o PostgreSQL

O serviço de gateway é hospedado em um grupo de nós de computação sem monitoração de estado situadas atrás de um endereço IP, o qual o cliente primeiro alcançaria ao tentar conectar-se a um servidor de Banco de

Dados do Azure para PostgreSQL.

Como parte da manutenção contínua do serviço, atualizaremos periodicamente o hardware de computação que hospeda os gateways para garantir a experiência mais segura e de alto desempenho. Quando o hardware de gateway for atualizado, um novo anel dos nós de computação será criado primeiro. Esse novo anel serve o tráfego para todos os servidores do banco de dados do Azure recém-criados para o PostgreSQL e ele terá um endereço IP diferente dos anéis de gateway mais antigos na mesma região para diferenciar o tráfego. O hardware de gateway mais antigo continua auxiliando os servidores existentes, mas será planejado para encerramento no futuro. Antes de desativar um hardware de gateway, os clientes que executam seus servidores e se conectam a anéis de gateway mais antigos serão notificados por e-mail e no portal do Azure três meses antes da desativação. A desativação dos gateways pode afetar a conectividade com seus servidores.

- Codificar os endereços IP do gateway na cadeia de conexão do seu aplicativo não é **recomendado**. Você deve usar o FQDN (nome de domínio totalmente qualificado) do servidor no formato.postgres.database.azure.com na cadeia de conexão do seu aplicativo.
- Você não atualiza os endereços IP de gateway mais recentes no firewall do lado do cliente a fim de permitir que o tráfego de saída seja capaz de alcançar nossos novos anéis de gateway.

A seguinte tabela lista os endereços IP do gateway do Banco de Dados do Azure para o gateway PostgreSQL para todas as regiões de dados. As informações dos endereços IP do gateway mais atualizadas para cada região são mantidas na tabela a seguir. Na tabela abaixo, a grade de colunas representa o seguinte:

- **Endereços IP do gateway:** esta coluna lista os endereços IP dos gateways atuais hospedados na última geração de hardware. Se você estiver provisionando um novo servidor, recomendamos que abra o firewall do lado do cliente a fim de permitir o tráfego de saída para os endereços IP listados nesta grade de coluna.
- **Endereços IP do gateway (desativação):** esta coluna lista os endereços IP dos gateways hospedados em uma geração mais antiga do hardware que está sendo desativada no momento. Se você estiver provisionando um novo servidor, poderá ignorar estes endereços IP. Se você tiver um servidor existente, continue a reter a regra de saída do firewall destes endereços IP, pois ainda não a desativamos. Se você remover as regras de firewall destes endereços IP, poderá ver erros de conectividade. Em vez disso, espera-se que você adicione proativamente os novos endereços IP listados na coluna de endereços IP do gateway, à regra de firewall de saída assim que receber a notificação de desativação. Isso garantirá que, quando o servidor for migrado para o hardware de gateway mais recente, não haja interrupções na conectividade com o servidor.
- **Endereços IP do gateway (desativado):** estas colunas listam os endereços IP dos anéis de gateway, que foram desativados e não estão mais em operação. Você pode remover com segurança estes endereços IP da regra de firewall de IP de saída.

DENOMINAÇÃO DA REGIÃO	ENDEREÇOS IP DO GATEWAY	ENDEREÇOS IP DO GATEWAY (DESATIVAÇÃO)	ENDEREÇOS IP DO GATEWAY (DESATIVAÇÃO)
Austrália Central	20.36.105.0		
Austrália Central2	20.36.113.0		
Leste da Austrália	13.75.149.87 - 40.79.161.1		
Sudeste da Austrália	191.239.192.109 - 13.73.109.251		
Brazil South	191.233.201.8 - 191.233.200.16		104.41.11.5
Canadá Central	40.85.224.249		

DENOMINAÇÃO DA REGIÃO	ENDEREÇOS IP DO GATEWAY	ENDEREÇOS IP DO GATEWAY (DESATIVAÇÃO)	ENDEREÇOS IP DO GATEWAY (DESATIVAÇÃO)
Leste do Canadá	40.86.226.166		
Centro dos EUA	23.99.160.139 - 52.182.136.37- 52.182.136.38	13.67.215.62	
Leste da China	139.219.130.35		
Leste da China 2	40.73.82.1		
Norte da China	139.219.15.17		
Norte da China 2	40.73.50.0		
Leste da Ásia	191.234.2.139 - 52.175.33.150 - 13.75.33.20 - 13.75.33.21		
Leste dos EUA	40.71.8.203 - 40.71.83.113	40.121.158.30	191.238.6.43
Leste dos EUA 2	40.70.144.38 - 52.167.105.38	52.177.185.181	
França Central	40.79.137.0 - 40.79.129.1		
Sul da França	40.79.177.0		
Alemanha Central	51.4.144.100		
Norte da Alemanha	51 -116.56.0		
Nordeste da Alemanha	51.5.144.179		
Centro-Oeste da Alemanha	51 -116.152.0		
Centro da Índia	104.211.96.159		
Sul da Índia	104.211.224.146		
Oeste da Índia	104.211.160.80		
Japan East	40.79.192.23 - 40.79.184.8	13.78.61.196	
Oeste do Japão	191.238.68.11 - 40.74.96.6 - 40.74.96.7	104.214.148.156	
Coreia Central	52.231.17.13	52.231.32.42	
Sul da Coreia	52.231.145.3	52.231.200.86	

DENOMINAÇÃO DA REGIÃO	ENDEREÇOS IP DO GATEWAY	ENDEREÇOS IP DO GATEWAY (DESATIVAÇÃO)	ENDEREÇOS IP DO GATEWAY (DESATIVAÇÃO)
Centro-Norte dos EUA	52.162.104.35 - 52.162.104.36	23.96.178.199	
Norte da Europa	52.138.224.6 - 52.138.224.7	40.113.93.91	191.235.193.75
Norte da África do Sul	102.133.152.0		
Oeste da África do Sul	102.133.24.0		
Centro-Sul dos Estados Unidos	104.214.16.39 - 20.45.120.0	13.66.62.124	23.98.162.75
Sudeste da Ásia	40.78.233.2 - 23.98.80.12	104.43.15.0	
Norte da Suíça	51.107.56.0		
Oeste da Suíça	51.107.152.0		
EAU Central	20.37.72.64		
Norte dos EAU	65.52.248.0		
Sul do Reino Unido	51.140.184.11		
Oeste do Reino Unido	51.141.8.11		
Centro-Oeste dos EUA	13.78.145.25		
Europa Ocidental	13.69.105.208 - 104.40.169.187	40.68.37.158	191.237.232.75
Oeste dos EUA	13.86.216.212 - 13.86.217.212	104.42.238.205	23.99.34.75
Oeste dos EUA 2	13.66.226.202		

Perguntas frequentes

O que você precisa saber acerca desta manutenção planejada?

Esta é apenas uma alteração de DNS que a torna transparente para os clientes. Embora o endereço IP do FQDN seja alterado no servidor DNS, o cache DNS local será atualizado em 5 minutos e isto será feito automaticamente pelos sistemas operacionais. Após a atualização do DNS local, todas as novas conexões se conectarão ao novo endereço IP; todas as conexões existentes permanecerão conectadas no endereço IP antigo sem interrupção até que os endereços IP antigos sejam completamente encerrados. O endereço IP antigo levará, aproximadamente, de três a quatro semanas para ser encerrado, ou seja, ele não deve ter nenhum efeito sobre os aplicativos cliente.

O que nós estamos desativando?

Somente os nós de gateway serão encerrados. Quando os usuários conectam-se a seus servidores, a primeira parada da conexão é no nó do gateway antes de a conexão ser encaminhada ao servidor. Estamos desativando os anéis de gateway antigos (não os anéis de locatário onde o servidor está em execução); consulte a [Arquitetura de conectividade](#) para obter mais esclarecimentos.

Como você pode validar se suas conexões irão para os nós de gateway antigos ou os novos nós de gateway?

Execute ping no FQDN do servidor, por exemplo `ping xxx.postgres.database.azure.com`. Se o endereço IP retornado for um dos IPs listados embaixo endereços IP do gateway (desativação) no documento acima, significa que sua conexão está passando pelo gateway antigo. Contrariamente, se o endereço IP retornado for um dos IPs listados nos endereços IP de gateway, significa que a conexão está passando pelo novo gateway.

Você também pode testar a conexão por [PSP](#) ou protocolo TCPPing com o servidor de banco de dados do seu aplicativo cliente pela porta 3306 e verificar se o endereço IP de retorno não é um dos endereços IP de desativação

Como fazer para saber quando a manutenção terminará e receberei outra notificação quando os endereços IP antigos forem encerrados?

Você receberá um e-mail para informá-lo quando iniciar o trabalho de manutenção. A manutenção pode demorar até um mês, dependendo do número de servidores que precisamos migrar em todas as regiões. Prepare seu cliente para conectar-se ao servidor de banco de dados usando o FQDN ou usando o novo endereço IP da tabela acima.

O que fazer se meus aplicativos cliente ainda estiverem conectando-se ao servidor de gateway antigo?

Isto indica que seus aplicativos conectam-se ao servidor usando o endereço IP estático em vez do FQDN. Revise as cadeias de conexão e a configuração do pool de conexões, a configuração do AKS ou até mesmo o código-fonte.

Existe algum impacto nas minhas conexões de aplicativo?

Essa manutenção é apenas uma alteração de DNS, portanto, é transparente para o cliente. Assim que o cache DNS é atualizado no cliente (feito automaticamente pelo sistema operacional), toda a nova conexão será feita com o novo endereço IP e toda a conexão existente funcionará bem até que o endereço IP antigo seja encerrado por completo, o que geralmente ocorre algumas semanas depois. E a lógica de aplicativo de repetição não é necessária nesse caso, mas é bom ver que o aplicativo tem a lógica de aplicativo de repetição configurada. Use o FQDN para conectar-se ao servidor de banco de dados ou habilite a lista de novos "endereços IP de gateway" na cadeia de conexão do aplicativo. Esta operação de manutenção não removerá as conexões existentes. Ela apenas marca as novas solicitações de conexão vão para o novo anel do gateway.

Posso solicitar uma janela de tempo específica da manutenção?

Como a migração deve ser transparente e não afeta a conectividade do cliente, esperamos que não haja nenhum problema para a maioria dos usuários. Revise seu aplicativo proativamente e certifique-se de usar o FQDN para conectar -se ao servidor de banco de dados ou habilitar a lista de novos "endereços IP de gateway" na cadeia de conexão do aplicativo.

Estou usando o link privado, minhas conexões serão afetadas?

Não, essa é uma desativação do hardware de gateway e não tem relação com link ou endereços IP privados; ela afetará apenas endereços IP públicos mencionados nos endereços IP de desativação.

Próximas etapas

- [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure](#)
- [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#)

Usar o Azure Active Directory para autenticação com o PostgreSQL

17/07/2021 • 6 minutes to read

A autenticação do Microsoft Azure Active Directory (Azure AD) é um mecanismo de conexão ao Banco de Dados do Azure para PostgreSQL usando identidades definidas no Azure AD. Com a autenticação do Azure AD, é possível gerenciar as identidades de usuários do banco de dados e outros serviços da Microsoft em uma única localização central, o que simplifica o gerenciamento de permissões.

Os benefícios de usar o Azure AD incluem:

- Autenticação de usuários em Serviços do Azure de forma uniforme
- Gerenciamento de políticas de senha e rotação de senhas em um único local
- Várias formas de autenticação com suporte pelo Azure Active Directory, o que pode eliminar a necessidade de armazenar senhas
- Os clientes podem gerenciar permissões de banco de dados usando grupos (Azure AD) externos.
- A autenticação do Azure AD usa funções de banco de dados PostgreSQL para autenticar identidades no banco de dados
- Suporte de autenticação baseada em token em aplicativos que se conectam ao Banco de Dados do Azure para PostgreSQL

Para configurar e usar a autenticação do Active Directory do Azure, use o seguinte processo:

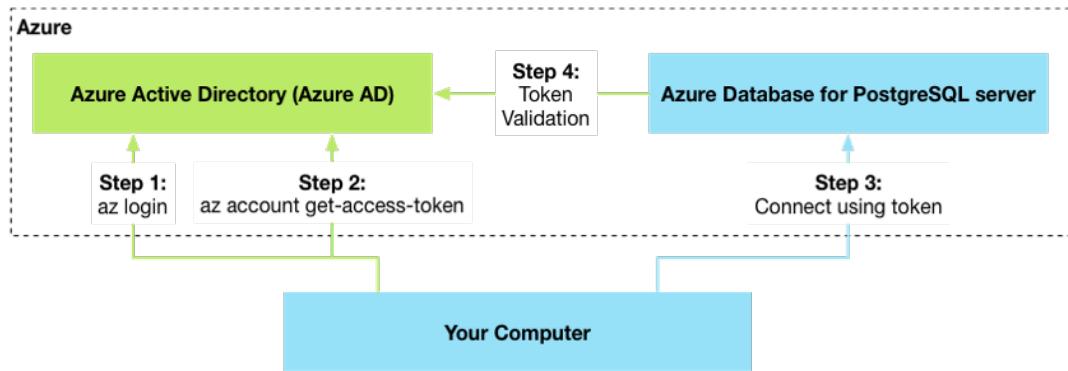
1. Crie e popule o Azure Active Directory com identidades de usuário, conforme necessário.
2. Opcionalmente, associe ou altere o Active Directory que está associado atualmente à sua assinatura do Azure.
3. Crie um administrador do Azure AD para o Banco de Dados do Azure para PostgreSQL.
4. Crie usuários de banco de dados em seu banco de dados, mapeados para identidades do Azure AD.
5. Conecte-se ao banco de dados recuperando um token para uma identidade do Azure AD e entrando.

NOTE

Para saber como criar e popular o Azure AD e configurar o Azure AD com o Banco de Dados do Azure para PostgreSQL, confira [Configurar e entrar com o Azure AD para Banco de Dados do Azure para PostgreSQL](#).

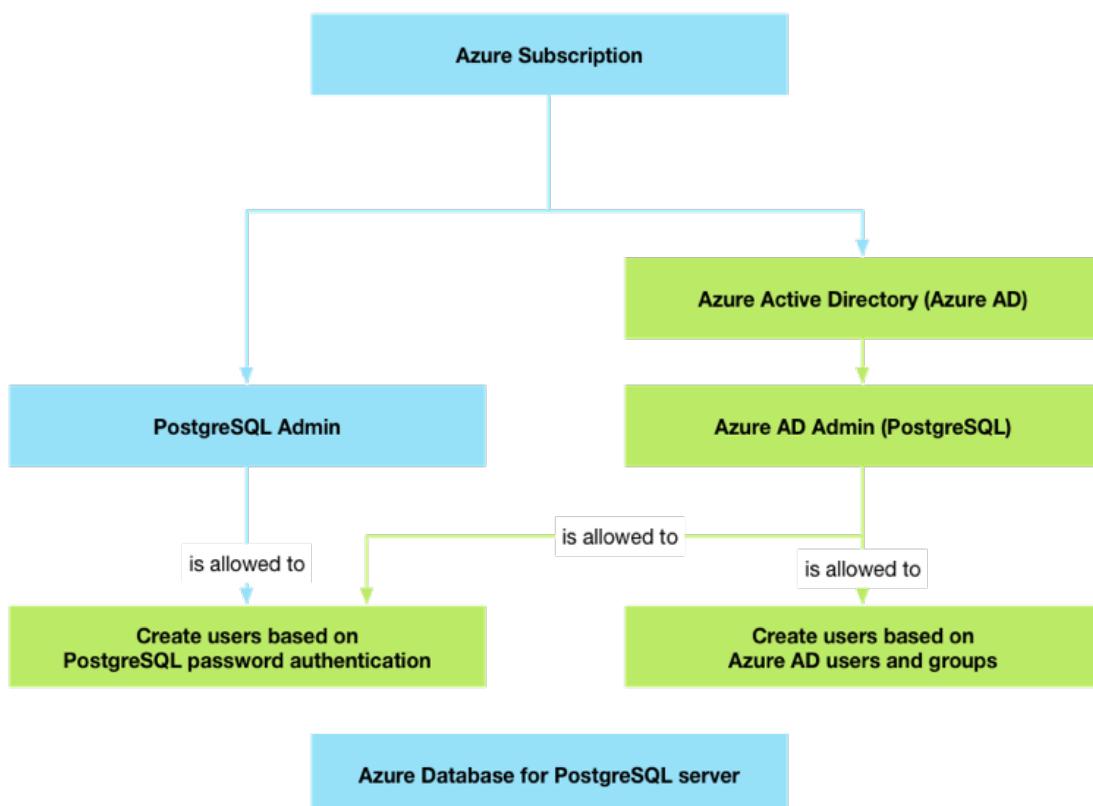
Arquitetura

O diagrama de alto nível a seguir resume como a autenticação funciona usando a autenticação do Azure Active Directory com o Banco de Dados do Azure para PostgreSQL. As setas indicam caminhos para comunicação.



Estrutura do administrador

Ao usar a autenticação do Azure AD, haverá duas contas de administrador para o servidor do PostgreSQL: o administrador original do PostgreSQL e o administrador do Azure AD. Somente o administrador com base em uma conta do AD do Azure pode criar o primeiro usuário de banco de dados do AD do Azure contido em um banco de dados de usuário. O logon de administrador do AD do Azure pode ser um usuário ou um grupo do AD do Azure. Quando o administrador é uma conta de grupo, ele pode ser usado por qualquer membro do grupo, permitindo múltiplos administradores do Azure AD para o servidor PostgreSQL. Usar a conta de grupo como um administrador aprimora a capacidade de gerenciamento, permitindo que você adicione e remova membros do grupo no Azure AD centralmente, sem alterar os usuários ou permissões no servidor PostgreSQL. Somente um administrador do AD do Azure (um usuário ou grupo) pode ser configurado por vez, a qualquer momento.



Permissões

Para criar novos usuários que podem se autenticar com o Azure AD, você deve ter a função `azure_ad_admin` no banco de dados. Essa função é atribuída ao configurar a conta de Administrador do Azure AD para um servidor de Banco de Dados do Azure para PostgreSQL.

Para criar um novo usuário de banco de dados do Azure AD, você deve se conectar como administrador do Azure AD. Isso é demonstrado em [Configurar e entrar com o Azure AD para o Banco de Dados do Azure para PostgreSQL](#).

Qualquer autenticação do Azure AD só será possível se o administrador do Azure AD tiver sido criado para o Banco de Dados do Azure para PostgreSQL. Se o administrador do Azure Active Directory tiver sido removido do servidor, os usuários existentes do Azure Active Directory criados anteriormente não poderão mais se conectar ao banco de dados usando suas credenciais do Azure Active Directory.

Conectar-se usando as identidades do Azure AD

A autenticação do Active Directory do Azure dá suporte aos seguintes métodos de conexão a um banco de dados usando identidades do AD do Azure:

- Senha do Azure Active Directory
- Integrada do Azure Active Directory
- Universal do Azure Active Directory com o MFA
- Usar certificados de aplicativos do Active Directory ou segredos de cliente
- [Identidade gerenciada](#)

Depois de estar autenticado no Active Directory, você recuperará um token. Esse token é sua senha para entrar.

Observe que as operações de gerenciamento, como adicionar novos usuários, têm suporte apenas para funções de usuário do Azure AD neste momento.

NOTE

Para obter mais detalhes sobre como se conectar com um token do Active Directory, confira [Configurar e entrar com o Azure AD para o Banco de Dados do Azure para PostgreSQL](#).

Considerações adicionais

- Para aumentar a capacidade de gerenciamento, é recomendável que você provisione um grupo dedicado do Microsoft Azure AD como administrador.
- Somente um administrador do Azure AD (um usuário ou grupo) pode ser configurado para um servidor do Banco de Dados do Azure para PostgreSQL a qualquer momento.
- Somente um administrador do Azure AD para PostgreSQL pode inicialmente se conectar ao Banco de Dados do Azure para PostgreSQL usando uma conta do Azure Active Directory. O administrador do Active Directory pode configurar os próximos usuários do banco de dados do Azure AD.
- Se um usuário for excluído do Azure AD, esse usuário não poderá mais se autenticar com o Azure AD e, portanto, não será mais possível adquirir um token de acesso para esse usuário. Nesse caso, embora a função correspondente ainda esteja no banco de dados, não será possível se conectar ao servidor com essa função.

NOTE

Ainda será possível entrar com o usuário do Azure AD excluído até que o token expire (até 60 minutos de emissão de tokens). Se você também remover o usuário do Banco de Dados do Azure para PostgreSQL, esse acesso será revogado imediatamente.

- Se o administrador do Azure AD for removido do servidor, o servidor não será mais associado a um locatário do Azure AD e, portanto, todos os logons do Azure AD serão desabilitados para o servidor. Adicione um novo administrador do Azure AD do mesmo locatário reabilitará logons do Azure AD.
- O Banco de Dados do Azure para PostgreSQL corresponde aos tokens de acesso para a função de Banco de Dados do Azure para PostgreSQL usando a ID de usuário do Azure AD exclusiva do usuário, em vez do nome de usuário. Isso significa que, se um usuário do Azure AD for excluído do Azure AD e um novo usuário for criado com o mesmo nome, o Banco de Dados do Azure para PostgreSQL o considerará um usuário diferente. Portanto, se um usuário for excluído do Azure AD e um novo usuário com o mesmo nome for adicionado, o novo usuário não poderá se conectar com a função existente. Para permitir isso, o administrador do Banco de Dados do Azure para PostgreSQL do Azure AD deverá revogar e, em seguida, conceder a função "azure_ad_user" ao usuário para atualizar a ID de usuário do Azure AD.

Próximas etapas

- Para saber como criar e popular o Azure AD e configurar o Azure AD com o Banco de Dados do Azure para PostgreSQL, confira [Configurar e entrar com o Azure AD para Banco de Dados do Azure para PostgreSQL](#).
- Para obter uma visão geral de logons, usuários e funções do Banco de Dados do Azure para PostgreSQL, confira [Criar usuários no Banco de Dados do Azure para PostgreSQL – Servidor Único](#).

Criptografia de dados de servidor único do Banco de Dados do Azure para PostgreSQL com uma chave gerenciada pelo cliente

21/05/2021 • 12 minutes to read

O PostgreSQL do Azure aproveita a [criptografia de armazenamento do Azure](#) para criptografar dados inativos por padrão, usando chaves gerenciadas pela Microsoft. Para usuários do PostgreSQL do Azure, é muito semelhante à TDE (Transparent Data Encryption) em outros bancos de dados, como o SQL Server. Muitas organizações exigem controle total sobre o acesso aos dados usando uma chave gerenciada pelo cliente. A criptografia de dados com chaves gerenciadas pelo cliente para o servidor único do Banco de Dados do Azure para PostgreSQL permite que você BYOK (Bring Your Own Key) para proteção de dados em repouso. Ela também permite que as organizações implementem a separação de tarefas no gerenciamento de chaves e dados. Com a criptografia gerenciada pelo cliente, você é responsável por (com controle total): ciclo de vida de uma chave, permissões de uso de chave e auditoria de operações em chaves.

A criptografia de dados com chaves gerenciadas pelo cliente para o servidor único do Banco de Dados do Azure para PostgreSQL é definida no nível do servidor. Para determinado servidor, uma chave gerenciada pelo cliente, chamada KEK (chave de criptografia de chave), é usada para criptografar a DEK (chave de criptografia de dados) usada pelo serviço. A KEK é uma chave assimétrica armazenada em uma instância do [Azure Key Vault](#) gerenciada pelo cliente e de propriedade dele. A KEK (chave de criptografia de chave) e a DEK (chave de criptografia de dados) são descritas em mais detalhes posteriormente neste artigo.

O Key Vault é um sistema de gerenciamento de chaves externas baseado em nuvem. Ele é altamente disponível e fornece armazenamento seguro escalonável para chaves de criptografia RSA, opcionalmente apoiado por HSMs (módulos de segurança de hardware) validados pelo FIPS 140-2 Nível 2. Ele não permite acesso direto a uma chave armazenada, mas fornece serviços de criptografia e descriptografia para as entidades autorizadas. O Key Vault pode gerar a chave, importá-la ou [transferi-la de um dispositivo HSM local](#).

NOTE

Esse recurso está disponível em todas as regiões do Azure nas quais o servidor único do Banco de Dados do Azure para PostgreSQL dá suporte aos tipos de preço "Uso Geral" e "Otimizado para Memória". Para obter outras limitações, veja a seção [Limitação](#).

Vantagens

A criptografia de dados com chaves gerenciadas pelo cliente para o servidor único do Banco de Dados do Azure para PostgreSQL oferece os benefícios a seguir:

- O acesso a dados é totalmente controlado por você pela capacidade de remover a chave e tornar o banco de dados inacessível
- Controle total sobre o ciclo de vida da chave, incluindo a rotação da chave para se alinhar com as políticas corporativas
- Gerenciamento central e organização de chaves no Azure Key Vault
- Capacidade de implementar a separação de tarefas entre os gerentes de segurança, DBA e administradores do sistema

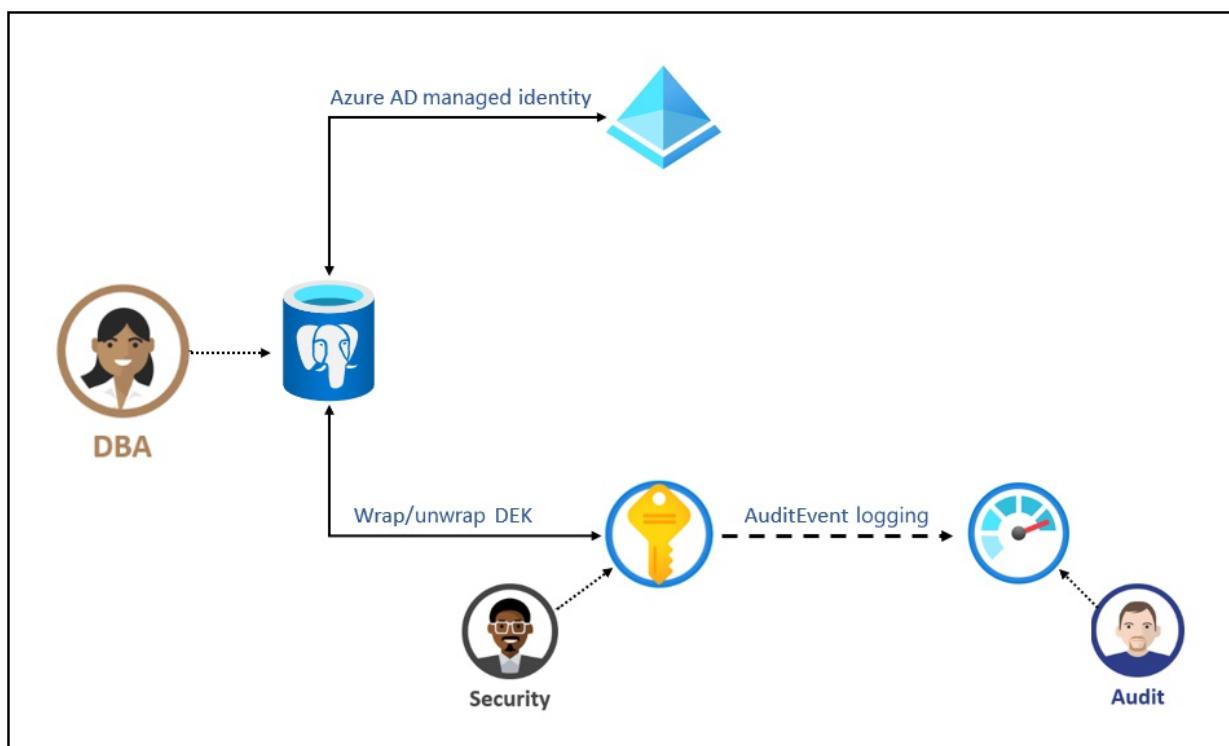
Descrição e terminologia

DEK (chave de criptografia de dados) : uma chave simétrica AES256 usada para criptografar uma partição ou bloco de dados. Criptografar cada bloco de dados com uma chave diferente torna os ataques de análise de criptografia mais difíceis. O acesso a DEKs é necessário pelo provedor de recursos ou instância do aplicativo que criptografa e descriptografa um bloco específico. Quando você substitui uma DEK por uma nova chave, apenas os dados no respectivo bloco associado precisam ser criptografados novamente com a nova chave.

KEK (chave de criptografia de chave) : uma chave de criptografia usada para criptografar as DEKs. Uma KEK que nunca deixa o Key Vault permite que as DEKs sejam criptografadas e controladas. A entidade que tem acesso à KEK pode ser diferente da entidade que requer a DEK. Uma vez que o KEK é necessário para descriptografar os DEKs, o KEK é efetivamente um ponto único pelo qual os DEKs podem ser efetivamente excluídos pela exclusão do KEK.

As DEKs, criptografadas com as KEKs, são armazenadas separadamente. Somente uma entidade com acesso à KEK pode descriptografar essas DEKs. Para obter mais informações, confira [Segurança na criptografia em repouso](#).

Como funciona a criptografia de dados com uma chave gerenciada pelo cliente



Para que um servidor PostgreSQL use chaves gerenciadas pelo cliente armazenadas no Key Vault para criptografia da DEK, um administrador do Key Vault fornece os seguintes direitos de acesso ao servidor:

- **obter**: para recuperar a parte pública e as propriedades da chave no cofre de chaves.
- **wrapKey**: para poder criptografar a DEK. O DEK criptografado é armazenado no Banco de Dados do Azure para PostgreSQL.
- **unwrapKey**: para poder descriptografar a DEK. O Banco de Dados do Azure para PostgreSQL precisa do DEK descriptografado para criptografar/descriptografar os dados

O administrador do cofre de chaves também pode [habilitar o registro em log de eventos de auditoria do Key Vault](#), para que eles possam ser auditados posteriormente.

Quando o servidor é configurado para usar a chave gerenciada pelo cliente armazenada no cofre de chaves, o

servidor envia a DEK para o cofre de chaves para criptografias. O Key Vault retorna a DEK criptografada, que é armazenada no banco de dados do usuário. De modo semelhante, quando necessário, o servidor envia a DEK protegida para o cofre de chaves para descriptografia. Os auditores poderão usar o Azure Monitor para examinar os logs de eventos do Key Vault se o registro em log estiver habilitado.

Requisitos para configurar a criptografia de dados para o servidor único do Banco de Dados do Azure para PostgreSQL

Veja a seguir os requisitos para configurar o Key Vault:

- O Key Vault e o servidor único do Banco de Dados do Azure para PostgreSQL precisam pertencer ao mesmo locatário do Azure AD (Azure Active Directory). Não há suporte para interações do servidor e do Key Vault entre locatários. A movimentação de recursos do Key Vault posteriormente exige que você reconfigure a criptografia de dados.
- O cofre de chaves deve ser definido com 90 dias para 'Dias para manter os cofres excluídos'. Se o cofre de chaves existente tiver sido configurado com um número mais baixo, você precisará criar um novo cofre de chaves, pois ele não pode ser modificado após a criação.
- Habilite o recurso de exclusão reversível no cofre de chaves para proteger contra perda de dados em caso de exclusão acidental da chave (ou do cofre de chaves). Os recursos excluídos com a exclusão reversível são retidos por 90 dias, a menos que o usuário os recupere ou limpe pelo. As ações de recuperação e limpeza têm suas próprias permissões associadas em uma política de acesso do Key Vault. O recurso de exclusão reversível está desativado por padrão, mas você pode habilitá-lo por meio do PowerShell ou da CLI do Azure (observe que não é possível habilitá-lo por meio do portal do Azure).
- Habilite a proteção contra limpeza para impor um período de retenção obrigatório para os cofres e os objetos de cofre excluídos
- Conceda o acesso de servidor único do Banco de Dados do Azure para PostgreSQL ao cofre de chaves com as permissões get, wrapKey e unwrapKey usando a identidade gerenciada exclusiva. No portal do Azure, a identidade exclusiva 'Serviço' é criada automaticamente quando a criptografia de dados é habilitada no servidor único PostgreSQL. Confira [Data encryption for Azure Database for PostgreSQL Single server by using the Azure portal](#) (Criptografia de dados para servidor único do Banco de Dados do Azure para PostgreSQL usando o portal do Azure) para obter instruções detalhadas passo a passo quando você estiver usando o portal do Azure.

Veja a seguir os requisitos para configurar a chave gerenciada pelo cliente:

- A chave gerenciada pelo cliente a ser usada para criptografar a DEK só pode ser assimétrica, RSA 2048.
- A data de ativação da chave (se definida) precisa ser uma data e uma hora no passado. A data de validade (se definida) precisa ser uma data e hora no futuro.
- A chave precisa estar no estado *Habilitado*.
- Se você estiver [importando uma chave existente](#) no cofre de chaves, certifique-se de fornecê-la nos formatos de arquivo com suporte (`.pfx` , `.byok` , `.backup`).

Recomendações

Quando você estiver usando a criptografia de dados usando uma chave gerenciada pelo cliente, veja as recomendações para configurar o Key Vault:

- Defina um bloqueio de recurso no Key Vault para controlar quem pode excluir esse recurso crítico e evitar a exclusão acidental ou não autorizada.
- Habilite a auditoria e relatórios em todas as chaves de criptografia. O Key Vault fornece logs que são fáceis de serem injetados em outras ferramentas de gerenciamento de eventos e informações de segurança. O Log Analytics do Azure Monitor é um exemplo de um serviço que já está integrado.

- Verifique se o Key Vault e o servidor único do Banco de Dados do Azure para PostgreSQL residem na mesma região, para garantir um acesso mais rápido para as operações de encapsulamento e desencapsulamento da DEK.
- Bloqueie o Azure Key Vault para apenas **ponto de extremidade privado** e **redes selecionadas** e permita somente serviços *confiáveis da Microsoft* para proteger os recursos.

Veja recomendações para configurar uma chave gerenciada pelo cliente:

- Mantenha uma cópia da chave gerenciada pelo cliente em um local seguro ou coloque-a no serviço de caução.
- Se o Key Vault gerar uma chave, crie um backup da chave antes de usá-la pela primeira vez. Você só pode restaurar o backup para o Key Vault. Para obter mais informações sobre o comando backup, confira [Backup-AzKeyVaultKey](#).

Condição de chave gerenciada pelo cliente inacessível

Quando você configura a criptografia de dados com uma chave gerenciada pelo cliente no Key Vault, o acesso contínuo a essa chave é necessário para que o servidor permaneça online. Se o servidor perder o acesso à chave gerenciada pelo cliente no Key Vault, o servidor começará a negar todas as conexões em dez minutos. O servidor emite uma mensagem de erro correspondente e altera o estado do servidor para *inacessível*. Algumas das razões pelas quais o servidor pode alcançar esse estado são:

- Se criarmos um servidor de restauração pontual para o servidor único do Banco de Dados do Azure para PostgreSQL, que tem criptografia de dados habilitada, o servidor recém-criado estará no estado *Inacessível*. Você pode corrigir o estado do servidor por meio do [portal do Azure](#) ou da [CLI](#).
- Se criarmos uma réplica de leitura para o servidor único do Banco de Dados do Azure para PostgreSQL, que tem criptografia de dados habilitada, o servidor de réplica estará no estado *Inacessível*. Você pode corrigir o estado do servidor por meio do [portal do Azure](#) ou da [CLI](#).
- Se você excluir o Key Vault, o servidor único do Banco de Dados do Azure para PostgreSQL não poderá acessar a chave e será movido para o estado *Inacessível*. Recupere o [Key Vault](#) e revalide a criptografia de dados para tornar o servidor *Disponível*.
- Se exclirmos a chave do Key Vault, o servidor único do Banco de Dados do Azure para PostgreSQL não poderá acessar a chave e será movido para o estado *Inacessível*. Recupere a [Chave](#) e revalide a criptografia de dados para tornar o servidor *Disponível*.
- Se a chave armazenada no Azure Key Vault expirar, ela se tornará inválida e o servidor único do Banco de Dados do Azure para PostgreSQL passará para o estado *Inacessível*. Estenda a data de expiração da chave usando a [CLI](#) e revalide a criptografia de dados para tornar o servidor *Disponível*.

Revogação accidental de acesso à chave do Key Vault

Pode acontecer de alguém com direitos de acesso suficientes ao Key Vault desabilitar accidentalmente o acesso do servidor à chave ao:

- Revogar as permissões get, wrapKey, and unwrapKey do cofre de chaves do servidor.
- Excluir a chave.
- Excluir o cofre de chaves.
- Alterar as regras de firewall do cofre de chaves.
- Excluir a identidade gerenciada do servidor no Azure AD.

Monitorar a chave gerenciada pelo cliente no Key Vault

Para monitorar o estado do banco de dados e habilitar o alerta para perda de acesso ao protetor do Transparent Data Encryption, configure os seguintes recursos do Azure:

- [Azure Resource Health](#): um banco de dados inacessível que perdeu o acesso à chave do cliente aparecerá como "Inacessível" após a negação da primeira conexão com o banco de dados.
- [Log de atividades](#): quando o acesso à chave do cliente falha no Key Vault gerenciado pelo cliente, as entradas são adicionadas ao log de atividades. Você poderá restabelecer o acesso assim que possível se criar alertas para esses eventos.
- [Grupos de ação](#): defina esses grupos para enviar notificações e alertas com base em suas preferências.

Restaurar e replicar com uma chave gerenciada pelo cliente no Key Vault

Depois que o servidor único do Banco de Dados do Azure para PostgreSQL é criptografado com uma chave gerenciada pelo cliente armazenada no Key Vault, qualquer cópia recém-criada do servidor também é criptografada. Você pode fazer essa nova cópia por meio de uma operação de restauração local ou geográfica, ou por meio de réplicas de leitura. No entanto, a cópia pode ser alterada para refletir a nova chave gerenciada pelo cliente para criptografia. Quando a chave gerenciada pelo cliente é alterada, os backups antigos do servidor começam a usar a chave mais recente.

Para evitar problemas durante a configuração da criptografia de dados gerenciados pelo cliente durante a restauração ou a criação de réplica de leitura, é importante seguir estas etapas nos servidores primário e restaurado/de réplica:

- Inicie o processo de restauração ou criação de réplica de leitura no servidor único primário do Banco de Dados do Azure para PostgreSQL.
- Mantenha o servidor recém-criado (restaurado/de réplica) em um estado inacessível, porque sua identidade exclusiva ainda não recebeu permissões para o Key Vault.
- No servidor restaurado/de réplica, revalide a chave gerenciada pelo cliente nas configurações de criptografia de dados. Isso garante que o servidor recém-criado receba permissões de encapsulamento e desencapsulamento para a chave armazenada no Key Vault.

Limitações

No Banco de Dados do Azure para PostgreSQL, o suporte para criptografia de dados inativos usando a CMK (chave gerenciada pelo cliente) tem poucas limitações -

- O suporte para essa funcionalidade é limitado para os tipos de preço **Uso Geral** e **Otimizado para Memória**.

- Esse recurso só tem suporte em regiões e servidores que dão suporte ao armazenamento de até 16TB. Para obter a lista de regiões do Azure que dão suporte ao armazenamento de até 16 TB, veja a seção de armazenamento na documentação [aqui](#)

NOTE

- Para todos os novos servidores PostgreSQL criados nas regiões listadas acima, o suporte para criptografia com as chaves do Gerenciador de clientes está **disponível**. O servidor PITR (restauração pontual) ou a réplica de leitura não está qualificada, embora teoricamente seja 'nova'.
- Para validar se o servidor provisionado dá suporte a até 16TB, você pode ir para a folha de tipo de preço no portal e ver o tamanho máximo de armazenamento compatível com o servidor provisionado. Caso você possa mover o controle deslizante até 4 TB, o servidor pode não dar suporte à criptografia com chaves gerenciadas pelo cliente. No entanto, os dados são criptografados usando chaves de serviço gerenciadas em todos os momentos. Se você tiver dúvidas, entre em contato com AskAzureDBforPostgreSQL@service.microsoft.com.

- A criptografia só tem suporte com a chave de criptografia RSA 2048.

Próximas etapas

Saiba como [configurar a criptografia de dados com uma chave gerenciada pelo cliente para o servidor único do Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#).

Banco de Dados do Azure para PostgreSQL

Criptografia dupla de infraestrutura

21/05/2021 • 4 minutes to read

O banco de dados do Azure para PostgreSQL usa a [criptografia de armazenamento de data em repouso](#) para dados usando chaves gerenciadas da Microsoft. Os dados, incluindo backups, são criptografados no disco e essa criptografia está sempre ativa e não pode ser desabilitada. A criptografia usa o módulo de criptografia do FIPS 140-2 validado e uma codificação de 256 de bits do AES para a criptografia de armazenamento do Azure.

A criptografia dupla de infraestrutura adiciona uma segunda camada de criptografia usando chaves gerenciadas pelo serviço. Ele usa o módulo de criptografia validado pelo FIPS 140-2, mas com um algoritmo de criptografia diferente. Isso fornece uma camada adicional de proteção para seus dados em repouso. A chave usada na criptografia dupla de infraestrutura também é gerenciada pelo serviço banco de dados do Azure para PostgreSQL. A criptografia dupla de infraestrutura não é habilitada por padrão, pois a camada adicional de criptografia pode ter um impacto no desempenho.

NOTE

Este recurso só tem suporte para os tipos de preço "Uso Geral" e "memória otimizada" no banco de dados do Azure para PostgreSQL.

A criptografia da camada de infraestrutura tem a vantagem de ser implementada na camada mais próxima do dispositivo de armazenamento ou dos cabos de rede. O banco de dados do Azure para PostgreSQL implementa as duas camadas de criptografia usando chaves gerenciadas pelo serviço. Embora ainda tecnicamente na camada de serviço, é muito próximo do hardware que armazena os dados em repouso. Opcionalmente, você ainda pode habilitar a criptografia de dados em repouso usando a [chave gerenciada pelo cliente](#) para o servidor PostgreSQL provisionado.

A implementação nas camadas de infraestrutura também dá suporte a uma diversidade de chaves. A infraestrutura deve estar ciente de diferentes clusters de máquinas e redes. Como tal, chaves diferentes são usadas para minimizar o raio de ataques de infraestrutura e uma variedade de falhas de hardware e rede.

NOTE

O uso da criptografia dupla de infraestrutura terá impacto sobre o desempenho Banco de Dados do Azure para PostgreSQL servidor devido ao processo de criptografia adicional.

Benefícios

A infraestrutura de criptografia dupla dados para o Banco de Dados do Azure para PostgreSQL fornece os seguintes benefícios:

1. **Diversidade adicional** de implementação de criptografia – a mudança planejada para a criptografia baseada em hardware vai diversificar ainda mais as implementações fornecendo uma implementação baseada em hardware, além da implementação baseada em software.
2. **Erros de implementação** – duas camadas de criptografia na camada de infraestrutura protegem contra erros no gerenciamento de cache ou de memória em camadas superiores que expõem dados de texto não criptografado. Além disso, as duas camadas também garantem erros na implementação da criptografia em geral.

A combinação deles fornece proteção forte contra ameaças comuns e pontos fracos usados para atacar a criptografia.

Cenários com suporte com criptografia dupla de infraestrutura

Os recursos de criptografia fornecidos pelo Banco de Dados do Azure para PostgreSQL podem ser usados juntos. Abaixo está um resumo dos vários cenários que você pode usar:

##	CRIPTOGRAFIA PADRÃO	CRIPTOGRAFIA DUPLA DE INFRAESTRUTURA	CRIPTOGRAFIA DE DADOS USANDO CHAVES GERENCIADAS PELO CLIENTE
1	<i>Sim</i>	<i>Não</i>	<i>Não</i>
2	<i>Sim</i>	<i>Sim</i>	<i>Não</i>
3	<i>Sim</i>	<i>Não</i>	<i>Sim</i>
4	<i>Sim</i>	<i>Sim</i>	<i>Sim</i>

IMPORTANT

- Cenários 2 e 4 terão impacto sobre o desempenho Banco de Dados do Azure para PostgreSQL servidor devido a camada de infraestrutura de criptografia adicional.
- Configurar a criptografia dupla de infraestrutura para Banco de Dados do Azure para PostgreSQL só é permitido durante a criação do servidor. Depois que o servidor for provisionado, você não poderá alterar a criptografia de armazenamento. No entanto, você ainda pode habilitar a Criptografia de dados usando chaves gerenciadas pelo cliente para o servidor criado com/sem criptografia dupla de infraestrutura.

Limitações

Por Banco de Dados do Azure para PostgreSQL, o suporte para criptografia dupla de infraestrutura usando a chave gerenciada pelo serviço tem as seguintes limitações:

- O suporte para essa funcionalidade é limitado para os tipos de preço **Uso Geral** e **Otimizado para Memória**.
- Esse recurso só tem suporte em regiões e servidores que dão suporte ao armazenamento de até 16 TB. Para obter a lista de regiões do Azure que dão suporte ao armazenamento de até 16 TB, consulte [documentação de armazenamento](#).

NOTE

- Para todos os **novos** servidores PostgreSQL criados nas regiões listadas acima, ainda o suporte de criptografia de dados com as chaves do Gerenciador de clientes. Nesse caso, os servidores criados por meio de PITR (restauração point-in-time) ou réplicas de leitura não se qualificam como "novos".
- Para validar se o servidor provisionado dá suporte a até 16 TB, você pode ir para a folha de preços no portal e ver se o controle deslizante de armazenamento pode ser movido para até 16 TB. Se você só pode mover o controle deslizante até 4 TB, o servidor pode não dar suporte à criptografia com chaves gerenciadas pelo cliente; no entanto, os dados são criptografados usando chaves gerenciadas pelo serviço em todos os momentos. Se você tiver dúvidas, entre em contato com AskAzureDBforPostgreSQL@service.microsoft.com.

Próximas etapas

Saiba como [configurar a criptografia dupla de infraestrutura para o Banco de Dados do Azure para PostgreSQL](#).

Controles de Conformidade Regulatória do Azure Policy para o Banco de Dados do Azure para PostgreSQL

13/08/2021 • 13 minutes to read

A [Conformidade Regulatória no Azure Policy](#) fornece definições de iniciativas criadas e gerenciadas pela Microsoft, conhecidas como *internos*, para os **domínios de conformidade** e os **controles de segurança** relacionados a diferentes padrões de conformidade. Esta página lista os **domínios de conformidade** e os **controles de segurança** para o Banco de Dados do Azure para PostgreSQL. Você pode atribuir os itens internos a um **controle de segurança** individualmente a fim de ajudar a manter seus recursos do Azure em conformidade com o padrão específico.

O título de cada definição de política interna leva à definição da política no portal do Azure. Use o link na coluna **Versão da Política** para ver a origem no [repositório GitHub do Azure Policy](#).

IMPORTANT

Cada controle abaixo está associado com uma ou mais definições do [Azure Policy](#). Essas políticas podem ajudar você a [avaliar a conformidade](#) com o controle. No entanto, geralmente não há uma correspondência um para um ou completa entre um controle e uma ou mais políticas. Dessa forma, **Conformidade** no Azure Policy refere-se somente às próprias políticas. Não garante que você está totalmente em conformidade com todos os requisitos de um controle. Além disso, o padrão de conformidade inclui controles que não são abordados por nenhuma definição do Azure Policy no momento. Portanto, a conformidade no Azure Policy é somente uma exibição parcial do status de conformidade geral. As associações entre os controles e as definições de Conformidade Regulatória do Azure Policy para esses padrões de conformidade podem ser alteradas ao longo do tempo.

Azure Security Benchmark

O [Azure Security Benchmark](#) fornece recomendações sobre como você pode proteger suas soluções de nuvem no Azure. Para ver como esse serviço é mapeado completamente para o Azure Security Benchmark, confira os [arquivos de mapeamento do Azure Security Benchmark](#).

Para examinar como as iniciativas internas disponíveis do Azure Policy de todos os serviços do Azure são mapeadas para esse padrão de conformidade, confira [Conformidade regulatória do Azure Policy – Azure Security Benchmark](#).

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA (PORTAL DO AZURE)	VERSÃO DA POLÍTICA (GITHUB)
Segurança de rede	NS-1	Implementar a segurança para tráfego interno	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSAO DA POLÍTICA
Segurança de rede	NS-2	Conectar redes privadas	O ponto de extremidade privado deve ser habilitado para servidores PostgreSQL	1.0.2
Segurança de rede	NS-3	Estabelecer o acesso à rede privada aos serviços do Azure	O ponto de extremidade privado deve ser habilitado para servidores PostgreSQL	1.0.2
Proteção de dados	DP-4	Criptografar as informações confidenciais em trânsito	'Importar conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Proteção de dados	DP-5	criptografar dados confidenciais em repouso	Os servidores PostgreSQL devem usar chaves gerenciadas pelo cliente para criptografar dados inativos	1.0.4
Backup e recuperação	BR-1	Garantir backups automatizados regulares	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1
Backup e recuperação	BR-2	Criptografar dados de backup	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1

Azure Security Benchmark v1

O [Azure Security Benchmark](#) fornece recomendações sobre como você pode proteger suas soluções de nuvem no Azure. Para ver como esse serviço é mapeado completamente para o Azure Security Benchmark, confira os [arquivos de mapeamento do Azure Security Benchmark](#).

Para examinar como as iniciativas internas disponíveis do Azure Policy de todos os serviços do Azure são mapeadas para esse padrão de conformidade, confira [Conformidade regulatória do Azure Policy – Azure Security Benchmark](#).

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA (PORTAL DO AZURE)	VERSÃO DA POLÍTICA (GITHUB)
Segurança de rede	1.1	proteger recursos usando grupos de segurança de rede ou o Firewall do Azure em sua Rede Virtual	O ponto de extremidade privado deve ser habilitado para servidores PostgreSQL	1.0.2
Proteção de dados	4.4	criptografar todas as informações confidenciais em trânsito	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Recuperação de dados	9.1	garantir backups automatizados regulares	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1
Recuperação de dados	9.2	realizar backups completos do sistema e fazer backup das chaves gerenciadas pelo cliente	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1

CIS Microsoft Azure Foundations Benchmark 1.1.0

Para examinar como as iniciativas internas disponíveis do Azure Policy de todos os serviços do Azure são mapeadas para esse padrão de conformidade, confira [Conformidade regulatória do Azure Policy – CIS Microsoft Azure Foundations Benchmark 1.1.0](#). Para saber mais sobre esse padrão de conformidade, confira [CIS Microsoft Azure Foundations Benchmark](#).

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA (PORTAL DO AZURE)	VERSÃO DA POLÍTICA (GITHUB)
Serviços de Banco de Dados	4.12	Garantir que o parâmetro de servidor 'log_checkpoints' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	Os pontos de verificação de log devem ser habilitados para os servidores de banco de dados PostgreSQL	1.0.0
Serviços de Banco de Dados	4.13	Garantir que a opção 'Impor conexão SSL' esteja definida para 'HABILITADO' para o servidor de banco de dados PostgreSQL	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Serviços de Banco de Dados	4.14	Garantir que o parâmetro de servidor 'log_connections' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	As conexões de log devem ser habilitadas para os servidores de banco de dados PostgreSQL	1.0.0
Serviços de Banco de Dados	4.15	Garantir que o parâmetro de servidor 'log_disconnections' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	As desconexões devem ser registradas em log para os servidores de banco de dados PostgreSQL.	1.0.0
Serviços de Banco de Dados	4.17	Garantir que o parâmetro de servidor 'connection_throttling' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	A limitação de conexão deve estar habilitada para os servidores de banco de dados PostgreSQL	1.0.0

CIS Microsoft Azure Foundations Benchmark 1.3.0

Para examinar como as iniciativas internas disponíveis no Azure Policy para todos os serviço do Azure são mapeadas desse padrão de conformidade, confira [Conformidade regulatória do Azure Policy – CIS Microsoft Azure Foundations Benchmark 1.3.0](#). Para saber mais sobre esse padrão de conformidade, confira [CIS Microsoft Azure Foundations Benchmark](#).

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA (PORTAL DO AZURE)	VERSÃO DA POLÍTICA (GITHUB)
Serviços de Banco de Dados	4.3.2	Garantir que a opção 'Impor conexão SSL' esteja definida para 'HABILITADO' para o servidor de banco de dados MySQL	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Serviços de Banco de Dados	4.3.3	Garantir que o parâmetro de servidor 'log_checkpoints' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	Os pontos de verificação de log devem ser habilitados para os servidores de banco de dados PostgreSQL	1.0.0

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Serviços de Banco de Dados	4.3.4	Garantir que o parâmetro de servidor 'log_connections' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	As conexões de log devem ser habilitadas para os servidores de banco de dados PostgreSQL	1.0.0
Serviços de Banco de Dados	4.3.5	Garantir que o parâmetro de servidor 'log_disconnections' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	As desconexões devem ser registradas em log para os servidores de banco de dados PostgreSQL.	1.0.0
Serviços de Banco de Dados	4.3.6	Garantir que o parâmetro de servidor 'connection_throttling' seja definido como 'ON' para o servidor de banco de dados PostgreSQL	A limitação de conexão deve estar habilitada para os servidores de banco de dados PostgreSQL	1.0.0

CMMC nível 3

Para examinar como as iniciativas internas disponíveis do Azure Policy de todos os serviços do Azure são mapeadas para esse padrão de conformidade, confira [Conformidade regulatória do Azure Policy – CMMC nível 3](#). Para saber mais sobre esse padrão de conformidade, confira [Cybersecurity Maturity Model Certification \(CMMC\)](#).

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA (PORTAL DO AZURE)	VERSÃO DA POLÍTICA (GITHUB)
Controle de acesso	AC.1.001	Limitar o acesso do sistema de informações a usuários autorizados, processos que atuam em nome de usuários autorizados e dispositivos (incluindo outros sistemas de informações).	O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	1.0.0

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Controle de acesso	AC.1.001	Limitar o acesso do sistema de informações a usuários autorizados, processos que atuam em nome de usuários autorizados e dispositivos (incluindo outros sistemas de informações).	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2
Controle de acesso	AC.1.002	Limitar o acesso do sistema de informações aos tipos de transações e funções que os usuários autorizados têm permissão para executar.	'Importar conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Controle de acesso	AC.1.002	Limitar o acesso do sistema de informações aos tipos de transações e funções que os usuários autorizados têm permissão para executar.	O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	1.0.0
Controle de acesso	AC.1.002	Limitar o acesso do sistema de informações aos tipos de transações e funções que os usuários autorizados têm permissão para executar.	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2
Controle de acesso	AC.2.016	Controle o fluxo de CUI de acordo com autorizações aprovadas.	O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	1.0.0
Controle de acesso	AC.2.016	Controle o fluxo de CUI de acordo com autorizações aprovadas.	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2
Gerenciamento de configuração	CM.3.068	Restrinja, desabilite ou impeça o uso de programas, funções, portas, protocolos e serviços não essenciais.	O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	1.0.0

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Gerenciamento de configuração	CM.3.068	Restrinja, desabilite ou impeça o uso de programas, funções, portas, protocolos e serviços não essenciais.	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2
Recuperação	RE.2.137	Executar e testar back-ups de dados regularmente.	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1
Recuperação	RE.3.139	Executar regularmente backups de dados completos, abrangentes e resilientes conforme definido pela organização.	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1
Proteção do Sistema e das Comunicações	SC.1.175	Monitore, controle e proteja as comunicações (ou seja, informações transmitidas ou recebidas por sistemas organizacionais) nos limites externos e os principais limites internos dos sistemas organizacionais.	O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	1.0.0
Proteção do Sistema e das Comunicações	SC.1.175	Monitore, controle e proteja as comunicações (ou seja, informações transmitidas ou recebidas por sistemas organizacionais) nos limites externos e os principais limites internos dos sistemas organizacionais.	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2
Proteção do Sistema e das Comunicações	SC.3.177	Empregar criptografia validada por FIPS quando usada para proteger a confidencialidade da CUI.	A criptografia de infraestrutura deve ser habilitada para servidores do Banco de Dados do Azure para PostgreSQL	1.0.0

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Proteção do Sistema e das Comunicações	SC.3.183	Negar o tráfego de comunicações de rede por padrão e permitir o tráfego de comunicações de rede por exceção (ou seja, negar tudo, permitir por exceção).	O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	1.0.0
Proteção do Sistema e das Comunicações	SC.3.183	Negar o tráfego de comunicações de rede por padrão e permitir o tráfego de comunicações de rede por exceção (ou seja, negar tudo, permitir por exceção).	O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	1.0.2
Proteção do Sistema e das Comunicações	SC.3.185	Implemente mecanismos criptográficos para impedir a divulgação não autorizada da CUI durante a transmissão, a menos que ela conte com proteções físicas alternativas.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Proteção do Sistema e das Comunicações	SC.3.190	Proteger a autenticidade das sessões de comunicação.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Proteção do Sistema e das Comunicações	SC.3.191	Proteja a confidencialidade do CUI em repouso.	A criptografia de infraestrutura deve ser habilitada para servidores do Banco de Dados do Azure para PostgreSQL	1.0.0

HIPAA HITRUST 9.2

Para examinar como as iniciativas internas disponíveis do Azure Policy de todos os serviços do Azure são mapeadas para esse padrão de conformidade, confira [Conformidade Regulatória do Azure Policy – HIPAA HITRUST 9.2](#). Para obter mais informações sobre esse padrão de conformidade, confira [HIPAA HITRUST 9.2](#).

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA (PORTAL DO AZURE)	VERSÃO DA POLÍTICA (GITHUB)
---------	----------------	--------------------	-------------------------------	--------------------------------

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Controle de conexão de rede	0809.01n2Organizacional.1234 – 01.n	O tráfego de rede é controlado de acordo com a política de controle de acesso da organização por meio do firewall e de outras restrições relacionadas à rede para cada ponto de acesso à rede ou interface gerenciada do serviço de telecomunicação externo.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Controle de conexão de rede	0810.01n2Organizacional.5 – 01.n	As informações transmitidas são protegidas e, no mínimo, criptografadas em redes abertas e públicas.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Controle de conexão de rede	0811.01n2Organizacional.6 – 01.n	As exceções à política de fluxo de tráfego são documentadas com uma missão/necessidade de suporte, pela duração da exceção, e examinadas pelo menos anualmente; as exceções da política de fluxo de tráfego são removidas quando não há mais suporte a uma missão/necessidade de negócios explícita.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Controle de conexão de rede	0812.01n2Organizacional.8 – 01.n	Dispositivos remotos que estabelecem uma conexão não remota não têm permissão para se comunicar com recursos externos (remotos).	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Controle de conexão de rede	0814.01n1Organizacional.12 – 01.n	A capacidade dos usuários de se conectar à rede interna é restrita usando uma política de negação por padrão e de permissão por exceção em interfaces gerenciadas de acordo com a política de controle de acesso e os requisitos de aplicativos clínicos e de negócios.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Identificação de riscos relacionados a partes externas	1450.05i2Organizacional.2 – 05.i	A organização obtém garantias satisfatórias de que há uma segurança razoável das informações na cadeia de fornecedores de informações executando uma revisão anual, que inclui todos os parceiros/provedores de terceiros dos quais a cadeia de fornecedores de informações depende.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1
Backup	1618.09l1Organizacional.45 – 09.l	Os backups são armazenados em uma localização fisicamente remota e segura, a uma distância suficiente para torná-los razoavelmente imunes contra danos aos dados no site primário, e os controles físicos e ambientais razoáveis estão em vigor para garantir a proteção na localização remoto.	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1

DOMÍNIO	ID DO CONTROLE	TÍTULO DO CONTROLE	POLÍTICA	VERSÃO DA POLÍTICA
Backup	1623.09l2Organizational.4 – 09.l	As informações sigilosas são copiadas em backup em um formato criptografado para garantir a confidencialidade.	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1
Backup	1626.09l3Organizational.5 – 09.l	A organização garante que uma cópia atual e recuperável das informações sigilosas esteja disponível antes da movimentação de servidores.	O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	1.0.1
Transações online	0947.09y2Organizational.2 – 09.y	A organização garante que o armazenamento dos detalhes da transação esteja localizado fora de qualquer ambiente publicamente acessível (por exemplo, em uma plataforma de armazenamento existente na intranet da organização) e não seja mantido nem exposto em um meio de armazenamento diretamente acessível pela Internet.	'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	1.0.1

Próximas etapas

- Saiba mais sobre a [Conformidade Regulatória do Azure Policy](#).
- Confira os internos no [repositório Azure Policy GitHub](#).

Regras de firewall no Banco de Dados do Azure para PostgreSQL – Servidor único

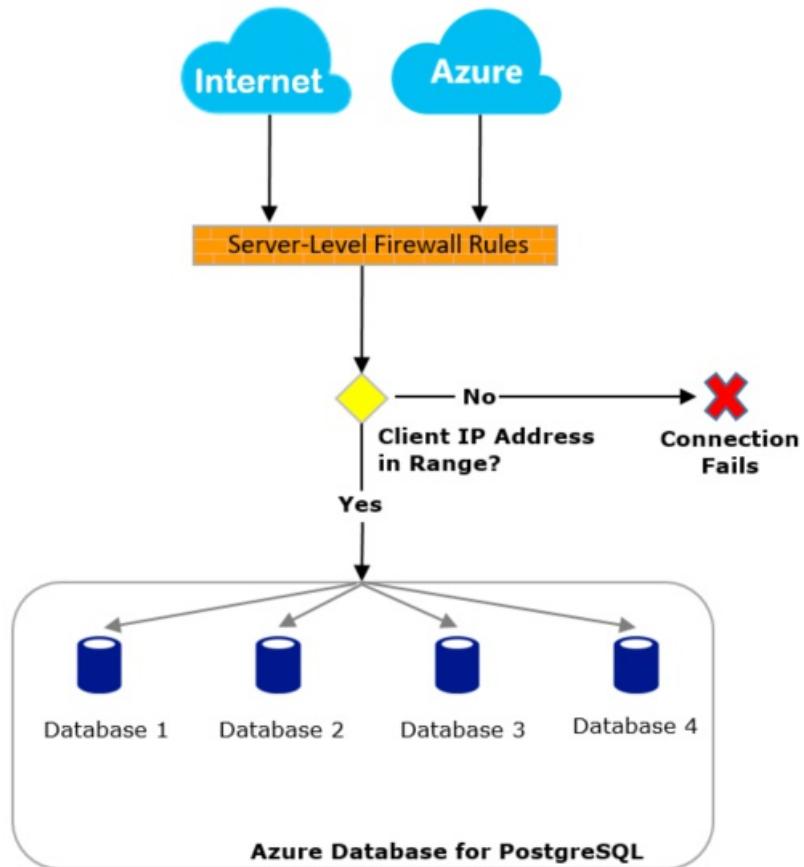
21/05/2021 • 5 minutes to read

O servidor do Banco de Dados do Azure para PostgreSQL é seguro por padrão, impedindo todo acesso ao servidor de banco de dados até que você especifique quais hosts de IP têm permissão para acessá-lo. O firewall concede acesso ao servidor com base no endereço IP de origem de cada solicitação. Para configurar seu firewall, você deve criar regras de firewall que especifiquem intervalos de endereços IP aceitáveis. Você pode criar regras de firewall no nível de servidor.

Regras de firewall: essas regras permitem que os clientes acessem todo o Banco de Dados do Azure para servidor PostgreSQL, ou seja, todos os bancos dentro do mesmo servidor lógico. As regras de firewall no nível de servidor podem ser configuradas por meio do Portal do Azure ou usando comandos da CLI do Azure. Para criar regras de firewall no nível de servidor, você deve ser o proprietário da assinatura ou um colaborador da assinatura.

Visão geral do firewall

Todo acesso ao servidor do Banco de Dados do Azure para PostgreSQL é bloqueado por padrão pelo firewall. Para acessar o servidor de outro computador/cliente ou aplicativo, especifique uma ou mais regras de firewall no nível do servidor para permitir o acesso ao servidor. Use as regras de firewall para especificar intervalos de endereços IP públicos permitidos. O acesso em si ao site do Portal do Azure não é afetado pelas regras de firewall. As tentativas de conexão da Internet e do Azure devem passar primeiramente pelo firewall antes de poderem acessar o Banco de Dados PostgreSQL, conforme exibido no diagrama a seguir:



Conectando pela Internet

As regras de firewall no nível do servidor se aplicam a todos os bancos de dados no mesmo servidor do Banco de Dados do Azure para PostgreSQL. Se o endereço IP fonte da solicitação estiver dentro de um dos intervalos especificados nas regras de firewall no nível do servidor, a conexão será concedida. Caso contrário, será rejeitada. Por exemplo, se o seu aplicativo se conectar ao driver JDBC para PostgreSQL, você poderá encontrar esse erro ao tentar se conectar quando o firewall estiver bloqueando a conexão.

```
java.util.concurrent.ExecutionException: java.lang.RuntimeException: org.postgresql.util.PSQLException:  
FATAL: no pg_hba.conf entry for host "123.45.67.890", user "adminuser", database "postgresql", SSL
```

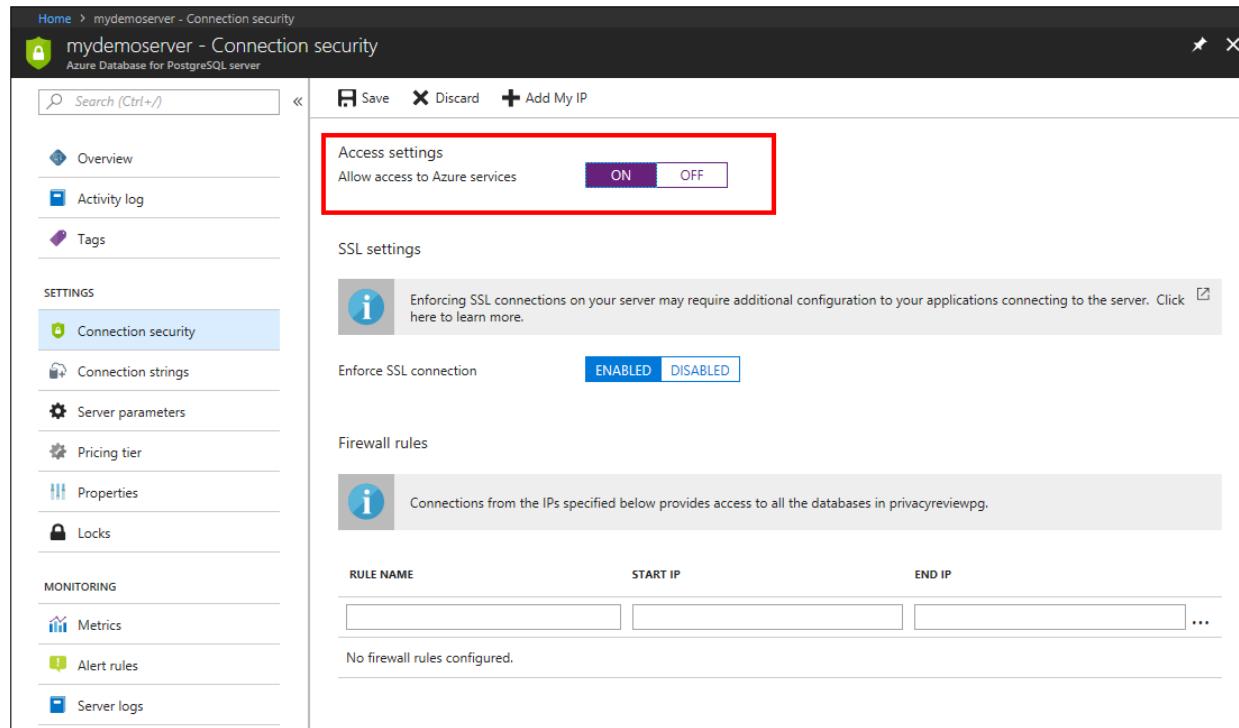
Conexão pelo Azure

É recomendável que você encontre o endereço IP de saída de um aplicativo ou serviço e permita explicitamente o acesso a esses endereços IP ou intervalos individuais. Por exemplo, você pode encontrar o endereço IP de saída de um Serviço de Aplicativo do Azure ou usar um IP público vinculado a uma máquina virtual ou outro recurso (veja abaixo para obter informações sobre como se conectar com o IP privado de uma máquina virtual nos pontos de extremidade de serviço).

Se um endereço IP de saída fixo não estiver disponível para o serviço do Azure, você poderá considerar habilitar conexões de todos os endereços IP do datacenter do Azure. Essa configuração pode ser habilitada no portal do Azure definindo a opção **Permitir acesso aos serviços do Azure** como **Ativado** no painel **Segurança de Conexão** e pressionando **Salvar**. Na CLI do Azure, uma configuração de regra de firewall com endereço inicial e final igual a 0.0.0.0 faz o equivalente. Se a tentativa de conexão for rejeitada pelas regras de firewall, ela não alcançará o servidor do Banco de Dados do Azure para PostgreSQL.

IMPORTANT

A opção **Permitir acesso aos serviços do Azure** configura o firewall para permitir todas as conexões do Azure, incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.



Conexão por uma VNet

Para se conectar com segurança ao servidor do Banco de Dados do Azure para PostgreSQL por uma VNet, considere o uso de [Pontos de extremidade de serviço de VNet](#).

Gerenciando programaticamente as regras de firewall

Além do Portal do Azure, as regras de firewall podem ser gerenciadas por meio de programação usando a CLI do Azure. Confira também [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#)

Solucionar problemas de firewall

Considere os seguintes pontos quando o acesso ao Banco de Dados do Microsoft Azure para o serviço de servidor PostgreSQL não se comportar conforme o esperado:

- **As alterações à lista de permissões ainda não entraram em vigor:** pode ocorrer um atraso de cinco minutos para que as alterações na configuração do firewall do Banco de Dados do Azure para servidor PostgreSQL entrem em vigor.
- **O logon não está autorizado ou uma senha incorreta foi usada:** se um logon não tiver permissões no servidor do Banco de Dados do Azure para servidor PostgreSQL, ou se a senha usada estiver incorreta, a conexão com o Banco de Dados do Azure para servidor PostgreSQL será negada. Criar uma configuração de firewall apenas oferece aos clientes uma oportunidade de tentar se conectar ao servidor; cada cliente ainda deverá fornecer as credenciais de segurança necessárias.

Por exemplo, usando um cliente JDBC, o seguinte erro pode aparecer.

```
java.util.concurrent.ExecutionException: java.lang.RuntimeException:  
  org.postgresql.util.PSQLException: FATAL: falha na autenticação da senha para o usuário  
  "seunomedesusário"
```

- **Endereço IP dinâmico:** se você tiver uma conexão com a Internet com endereçamento IP dinâmico e estiver com dificuldades para atravessar o firewall, tente uma das seguintes soluções:
 - Peça ao seu Provedor de serviços de Internet (ISP) o intervalo de endereços IP atribuído aos computadores clientes que acessarão o servidor de Banco de Dados do Azure para servidor PostgreSQL e, em seguida, adicione o intervalo de endereços IP como uma regra de firewall.
 - Obtenha o endereçamento IP estático para os computadores cliente e adicione os endereços IP estáticos como uma regra de firewall.
- **O IP do servidor parece ser público:** as conexões com o servidor do Banco de Dados do Azure para PostgreSQL são roteadas por meio de um gateway do Azure acessível publicamente. No entanto, o IP do servidor real é protegido pelo firewall. Para obter mais informações, consulte o [artigo sobre arquitetura de conectividade](#).
- **Não é possível se conectar do recurso do Azure com o IP permitido:** verifique se o ponto de extremidade de serviço **Microsoft.Sql** está habilitado para a sub-rede da qual você está se conectando. Se o **Microsoft.Sql** estiver habilitado, ele indicará que você só deseja usar [Regras de ponto de extremidade de serviço de VNet](#) nessa sub-rede.

Por exemplo, você poderá ver o erro a seguir se estiver se conectando de uma VM do Azure em uma sub-rede que tem o **Microsoft.Sql** habilitado, mas não tem nenhuma regra de VNet correspondente:

```
FATAL: Client from Azure Virtual Networks is not allowed to access the server
```

- **A regra de firewall não está disponível para o formato IPv6:** as regras de firewall devem estar no formato IPv4. Se você especificar regras de firewall no formato IPv6, ele mostrará o erro de validação.

Próximas etapas

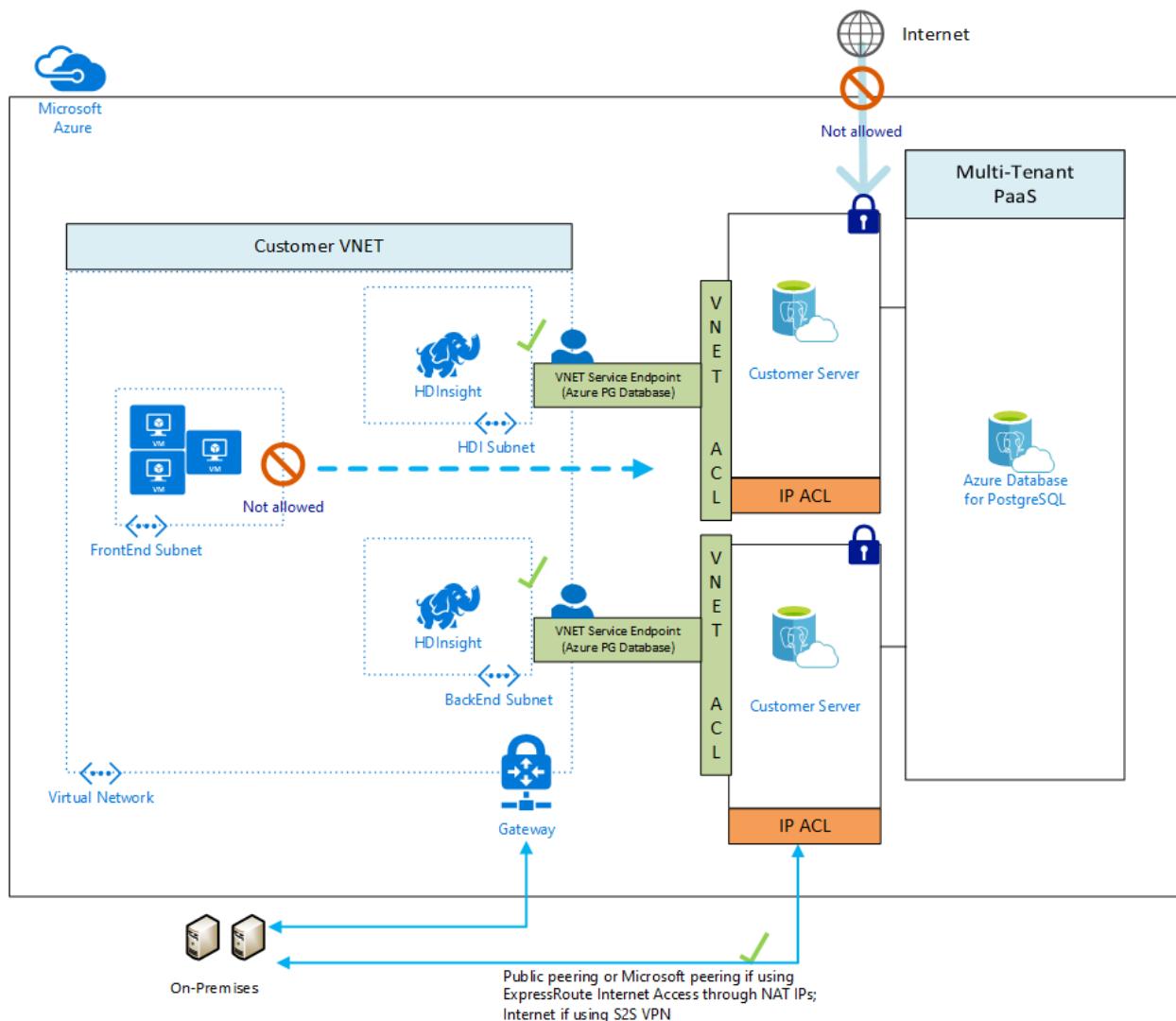
- [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure](#)
- [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#)
- [Pontos de extremidade de serviço da VNet no Banco de Dados do Azure para PostgreSQL](#)

Usar regras e pontos de extremidade de serviço de Rede Virtual para o Banco de Dados do Azure para PostgreSQL – Servidor único

09/08/2021 • 9 minutes to read

As *regras da rede virtual* são um recurso de segurança de firewall que controla se o servidor do Banco de Dados do Azure para PostgreSQL aceita comunicações que sejam enviadas de sub-redes particulares em redes virtuais. Este artigo explica por que o recurso de regra da rede virtual é, às vezes, a melhor opção para permitir a comunicação segura com seu servidor do Banco de Dados do Azure para PostgreSQL.

Para criar uma regra da rede virtual, deve haver primeiro uma VNet ([rede virtual](#)) e um [ponto de extremidade de serviço de rede virtual](#) para a regra a ser referenciada. A figura a seguir ilustra como um ponto de extremidade de serviço de Rede virtual funciona com o Banco de Dados do Azure para PostgreSQL:



NOTE

Esse recurso está disponível em todas as regiões da nuvem pública do Azure nas quais o Banco de Dados do Azure para PostgreSQL é implantado para servidores de Finalidade Geral e Otimizado por Memória. No caso de emparelhamento de VNet, se o tráfego estiver fluindo através de um Gateway VNet comum com pontos de extremidade de serviço e deve fluir para o par, crie uma regra ACL/VNet para permitir que Máquinas Virtuais do Azure acessem o Banco de Dados do Azure para servidor PostgreSQL.

Considere também o uso do [Link Privado](#) para conexões. O Link Privado fornece um endereço IP privado em sua VNet para o servidor do Banco de Dados do Azure para PostgreSQL.

Descrição e terminologia

Rede virtual: você pode ter redes virtuais associadas à sua assinatura do Azure.

Sub-rede: uma rede virtual contém **sub-redes**. Todas as VMs (máquinas virtuais) do Azure na VNet são atribuídas a uma sub-rede. Uma sub-rede pode conter várias VMs e/ou outros nós de computação. Nós de computadores que estão fora da sua rede virtual não podem acessar sua rede virtual, a menos que você configure a segurança para permitir o acesso.

Ponto de extremidade de serviço de Rede Virtual: Um [ponto de extremidade de serviço de Rede Virtual](#) é uma sub-rede cujos valores de propriedade incluem um ou mais nomes formais de tipo de serviço do Azure. Neste artigo, estamos interessados no nome do tipo de **Microsoft.Sql**, que faz referência ao serviço do Azure chamado banco de dados SQL. Essa marcação de serviço também aplica-se aos serviços MySQL e Banco de Dados do Azure para PostgreSQL. É importante observar que, ao aplicar a marca de serviço **Microsoft.Sql** a um ponto de extremidade de serviço de VNet, ela configurará o tráfego do ponto de extremidade para todos os servidores do Banco de Dados SQL do Azure, Azure Synapse Analytics, Banco de Dados do Azure para PostgreSQL e Banco de Dados do Azure para MySQL na sub-rede.

Regra da rede virtual: uma regra da rede virtual para seu servidor do Banco de Dados do Azure para PostgreSQL é uma sub-rede listada no ACL (lista de controle de acesso) do seu servidor do Banco de Dados do Azure para PostgreSQL. Para estar no ACL do seu servidor do Banco de Dados do Azure para PostgreSQL, a sub-rede deve conter o nome do tipo **Microsoft.Sql**.

Uma regra da rede virtual instrui o servidor do Banco de Dados do Azure para PostgreSQL a aceitar comunicações de cada nó na sub-rede.

Benefícios de uma regra da rede virtual

Até que você execute uma ação, as VMs em suas sub-redes não podem se comunicar com seu servidor do Banco de Dados do Azure para PostgreSQL. Uma ação que estabelece a comunicação é a criação de uma regra da rede virtual. A lógica para escolher a abordagem de regra da VNet requer uma discussão de comparação e contraste que envolve as opções de segurança concorrentes oferecidas pelo firewall.

Permitir acesso aos serviços do Azure

O painel de segurança de conexão tem um botão de **ON/OFF** rotulado como **Permitir acesso aos serviços do Azure**. A configuração **ON** permite as comunicações de todos os endereços IP do Azure e todas as sub-redes do Azure. Esses IPs ou sub-redes do Azure não podem pertencer a você. Essa configuração **ON** é provavelmente mais aberta do que você deseja que seu Banco de Dados do Azure para PostgreSQL seja. O recurso de regra da rede virtual oferece um maior controle granular.

Regras de IP

O firewall do Banco de Dados do Azure para PostgreSQL permite que você especifique intervalos de endereços

IP dos quais as comunicações são aceitas no Banco de Dados do Azure para PostgreSQL. Essa abordagem é adequada para endereços IP estáveis que estão fora da rede privada do Azure. Mas muitos nós dentro da rede privada do Azure estão configurados com endereços IP *dinâmicos*. Endereços IP dinâmicos podem mudar, como quando sua VM é reiniciada. Seria ilusório especificar um endereço IP dinâmico em uma regra de firewall, em um ambiente de produção.

Você pode recuperar a opção de IP obtendo um endereço IP *estático* para a VM. Para obter mais detalhes, consulte [Configurar endereços IP particulares para uma máquina virtual usando o Portal do Azure](#).

No entanto, a abordagem de IP estático pode se tornar difícil de gerenciar e é cara quando realizada em escala. Regras da rede virtual são mais fáceis de estabelecer e gerenciar.

Detalhes sobre as regras da rede virtual

Esta seção descreve vários detalhes sobre as regras da rede virtual.

Apenas uma região geográfica

Cada ponto de extremidade de serviço de rede virtual se aplica a apenas uma região do Azure. O ponto de extremidade não permite que outras regiões aceitem a comunicação da sub-rede.

Qualquer regra da rede virtual é limitada à região à qual seu ponto de extremidade subjacente se aplica.

Nível de servidor, não o nível de banco de dados

Cada regra da rede virtual aplica-se a todo o seu servidor do Banco de Dados do Azure para PostgreSQL, não apenas a um determinado banco de dados no servidor. Em outras palavras, a regra da rede virtual se aplica no nível de servidor, não no nível do banco de dados.

Funções de administração de segurança

Há uma separação de funções de segurança na administração de pontos de extremidade de serviço de rede virtual. A ação é necessária em cada uma das seguintes funções:

- **Administrador de rede:** ativar o ponto de extremidade.
- **Administrador de banco de dados:** atualize a ACL (lista de controle de acesso) para adicionar a sub-rede fornecida ao servidor do Banco de Dados do Azure para PostgreSQL.

Alternativa ao RBAC do Azure:

As funções de Administrador de banco de dados e Administrador de rede têm mais recursos do que o necessário para gerenciar regras da rede virtual. É necessário apenas um subconjunto de seus recursos.

No Azure, você tem a opção de usar o [RBAC do Azure \(Controle de acesso baseado em função\)](#) para criar uma única função personalizada com apenas o subconjunto necessário de recursos. Em vez de envolver o Administrador da Rede ou o do Banco de Dados, você pode usar a função personalizada. A área de superfície de sua exposição de segurança será menor se você adicionar um usuário a uma função personalizada em vez de adicioná-lo às outras duas funções principais de administrador.

NOTE

Em alguns casos, o Banco de Dados do Azure para PostgreSQL e a sub-rede da VNet estão em assinaturas diferentes. Nesses casos, você deve garantir as seguintes configurações:

- Ambas as assinaturas devem estar associadas ao mesmo locatário do Azure Active Directory.
- O usuário tem as permissões necessárias para iniciar operações, como habilitar pontos de extremidade de serviço e adicionar uma sub-rede da VNet ao servidor.
- Verifique se a assinatura tem os provedores de recursos **Microsoft.Sql** e **Microsoft.DBforPostgreSQL** registrados. Para obter mais informações, confira [resource-manager-registration](#)

Limitações

Para o Banco de Dados do Azure para PostgreSQL, o recurso de regras da rede virtual tem as seguintes limitações:

- Um aplicativo Web pode ser mapeado para um IP privado em uma sub-rede/rede virtual. Mesmo se os pontos de extremidade de serviço são ativados por meio da rede virtual/sub-rede determinada, as conexões do aplicativo Web para o servidor terão uma fonte IP pública Azure, não uma fonte de sub-rede/rede virtual. Para habilitar a conectividade de um aplicativo Web para um servidor com regras de firewall de VNET, você precisa Permitir que os serviços do Azure acessem o servidor no servidor.
- No firewall do Banco de Dados do Azure para PostgreSQL, cada regra da rede virtual referencia uma sub-rede. Todas essas sub-redes referenciadas devem ser hospedadas na mesma região geográfica que hospeda o Banco de Dados do Azure para PostgreSQL.
- Cada servidor do Banco de Dados do Azure para PostgreSQL pode ter até 128 entradas de ACL para qualquer rede virtual especificada.
- As regras da rede virtual se aplicam somente a redes virtuais do Azure Resource Manager; e não a redes do [modelo de implantação clássico](#).
- Ativar pontos de extremidade de serviço de rede virtual para o Banco de Dados do Azure para PostgreSQL usando a marca de serviço **Microsoft.Sql** também habilita os pontos de extremidade para todos os serviços de Banco de Dados do Azure: Banco de Dados do Azure para MySQL, Banco de Dados do Azure para PostgreSQL, Banco de Dados SQL do Azure e Azure Synapse Analytics.
- O suporte para ponto de extremidade de serviço de VNet é apenas para servidores de Uso Geral e Otimizados para Memória.
- Se o **Microsoft.Sql** estiver habilitado em uma sub-rede, isso indicará que você só deseja usar regras de VNet para se conectar. [Regras de firewall não VNet](#) de recursos nessa sub-rede não funcionarão.
- No firewall, os intervalos de endereços IP se aplicam aos seguintes itens de rede, mas as regras da rede virtual não:
 - [Rede privada virtual \(VPN\) de site a site \(S2S\)](#)
 - No local por meio de [ExpressRoute](#)

ExpressRoute

Se sua rede estiver conectada à rede do Azure através do [ExpressRoute](#), cada circuito é configurado com dois endereços IP públicos no Microsoft Edge. Os dois endereços IP são usados para se conectar aos serviços da Microsoft, como o Armazenamento do Azure, usando o emparelhamento público do Azure.

Para permitir a comunicação do seu circuito com o Banco de Dados do Azure para PostgreSQL, é necessário criar regras de rede IP para os endereços IP públicos dos seus circuitos. Para localizar os endereços IP públicos do seu circuito do ExpressRoute, abra um tique de suporte com o ExpressRoute por meio do Portal do Azure.

Adicionar uma regra do Firewall VNet ao seu servidor sem ativar os pontos de extremidade do serviço de VNet

Simplesmente definir uma regra de firewall VNet não ajuda a proteger o servidor para a VNet. Você também deve **ativar** os pontos de extremidade de serviço da VNet para que a segurança entre em vigor. Quando você ativa endpoints de serviço, sua sub-rede VNet passa por um período de inatividade até concluir a transição de **Desativado** para **Ativado**. Isso é especialmente verdadeiro no contexto de VNETs grandes. Use o sinalizador **IgnoreMissingServiceEndpoint** para reduzir ou eliminar o tempo de inatividade durante a transição.

Você pode definir o sinalizador `IgnoreMissingServiceEndpoint` usando a CLI do Azure ou o portal.

Artigos relacionados

- [Redes virtuais do Azure](#)
- [Pontos de extremidade de serviço de rede virtual do Azure](#)

Próximas etapas

Para obter artigos sobre como criar regras de VNet, consulte:

- [Criar e gerenciar regras de VNet do Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#)
- [Criar e gerenciar regras de VNet do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#)

Link Privado para o Banco de Dados do Azure para PostgreSQL - Servidor único

21/05/2021 • 8 minutes to read

O link privado permite criar pontos de extremidade privados para o Banco de Dados PostgreSQL do Azure - Servidor único a fim de trazê-lo para dentro da sua Rede Virtual (VNet). O ponto de extremidade privado expõe um IP privado em uma sub-rede que você pode usar para se conectar ao seu servidor de banco de dados, assim como qualquer outro recurso na VNet.

Para obter uma lista dos serviços de PaaS que permitem a funcionalidade do Link Privado, leia a [documentação](#) do Link Privado. Um ponto de extremidade privado é um endereço IP privado em uma [VNET](#) e sub-rede específicas.

NOTE

O recurso de link privado está disponível apenas para servidores do Banco de Dados do Azure para PostgreSQL nas camadas de preços de Uso geral ou Otimizado para Memória. Certifique-se de que o servidor de banco de dados esteja em um desses níveis de preços.

Prevenção contra exportação de dados

A exportação de dados no Banco de Dados do Azure para PostgreSQL ocorre quando um usuário autorizado, como um administrador de banco de dados, tem a capacidade de extrair dados de um sistema e movê-los para outra localização ou outro sistema fora da organização. Por exemplo, o usuário move os dados para uma conta de armazenamento pertencente a terceiros.

Considere um cenário com um usuário que executa o PGAdmin dentro de uma VM (máquina virtual) do Azure que está se conectando a um Banco de Dados do Azure para PostgreSQL - Servidor único provisionado no Oeste dos EUA. O exemplo abaixo mostra como limitar o acesso com pontos de extremidade públicos no servidor único do Banco de Dados do Azure para PostgreSQL usando controles de acesso à rede.

- Desabilite todo o tráfego de serviço do Azure para o Banco de Dados do Azure para PostgreSQL por meio do ponto de extremidade público. Para isso, configure *Permitir Serviços do Azure* como DESATIVADO. Certifique-se de que nenhum endereço IP ou intervalo tenha permissão para acessar o servidor por meio de [regras de firewall](#) ou [pontos de extremidade de serviço de rede virtual](#).
- Só permita o tráfego para o Banco de Dados do Azure para PostgreSQL que usa o endereço IP privado da VM. Para obter mais informações, confira os artigos sobre o [ponto de extremidade de serviço](#) e as [regras de firewall da VNet](#).
- Na VM do Azure, restrinja o escopo da conexão de saída usando NSGs (grupos de segurança de rede) e Marcas de Serviço, conforme mostrado a seguir
 - Especifique uma regra NSG para permitir tráfego para *Tag de serviço = SQL.WestUS* - só permitindo a conexão ao Banco de Dados do Azure para PostgreSQL no Oeste dos EUA
 - Especifique uma regra NSG (com uma prioridade mais alta) para negar o tráfego para a *Tag de Serviço = SQL* - negando as conexões com o Banco de Dados PostgreSQL em todas as regiões

Ao final desta configuração, a VM do Azure só poderá se conectar ao Banco de Dados do Azure para PostgreSQL - Servidor único na região Oeste dos EUA. No entanto, a conectividade não é restrita a um único Banco de

Dados do Azure para PostgreSQL - Servidor único. A VM ainda pode se conectar a qualquer Banco de Dados do Azure para PostgreSQL - Servidor único na região Oeste dos EUA, incluindo os bancos de dados que não fazem parte da assinatura. Embora tenhamos reduzido o escopo da exportação de dados no cenário acima a uma região específica, não o eliminamos por completo.

Agora, com o Link Privado, é possível configurar controles de acesso à rede, como NSGs, para restringir o acesso ao ponto de extremidade privado. Os recursos de PaaS individuais do Azure são então mapeados para pontos de extremidade privados específicos. Um usuário interno mal-intencionado só pode acessar o recurso de PaaS mapeado (por exemplo, um Banco de Dados do Azure para PostgreSQL - Servidor único) e nenhum outro recurso.

Conectividade local no emparelhamento privado

Quando você se conecta ao ponto de extremidade público a partir de computadores locais, seu endereço IP precisa ser adicionado ao firewall baseado em IP usando uma regra de firewall no nível de servidor. Embora esse modelo funcione bem para permitir o acesso a computadores individuais para cargas de trabalho de desenvolvimento ou teste, é difícil gerenciá-lo em um ambiente de produção.

Com o Link Privado, você pode habilitar o acesso entre instalações ao ponto de extremidade privado usando o [Express Route](#) (ER), o emparelhamento privado ou o [túnel de VPN](#). Em seguida, eles poderão desabilitar todo o acesso por meio do ponto de extremidade público e não usar o firewall baseado em IP.

NOTE

Em alguns casos, o Banco de Dados do Azure para PostgreSQL e a sub-rede da VNet estão em assinaturas diferentes. Nesses casos, você deve garantir as seguintes configurações:

- Verifique se as duas assinaturas têm o provedor de recursos `Microsoft.DBforPostgreSQL` registrado. Para obter mais informações, confira [resource-manager-registration](#)

Configurar o Link Privado para o Banco de Dados do Azure para PostgreSQL - Servidor único

Processo de criação

Pontos de extremidade privados são necessários para habilitar o Link Privado. Isso pode ser feito usando os seguintes guias de instruções.

- [Azure portal](#)
- [CLI](#)

Processo de aprovação

Depois que o administrador de rede criar o ponto de extremidade privado (PE), o administrador do PostgreSQL poderá gerenciar a conexão do ponto de extremidade privado (PEC) com o Banco de Dados do Azure para PostgreSQL. Essa separação de tarefas entre o administrador de rede e o DBA é útil para o gerenciamento da conectividade do Banco de Dados do Azure para PostgreSQL.

- Navegue até o recurso do servidor do Banco de Dados do Azure para PostgreSQL no portal do Azure.
 - Selecione as conexões de ponto de extremidade privado no painel esquerdo
 - Mostra uma lista de todas as PECs (conexões de ponto de extremidade privado)
 - Ponto de extremidade privado correspondente (PE) criado

Connection name	State	Private endpoint name	Request/Response Message
demopostgresqlprivatelink-499c76ef-86da-44f7-bde7-da...	Approved	demopostgresqlprivatelink	Auto-approved

- Selecione uma PEC individual na lista selecionando-a.

CONNECTION NAME	STATE	PRIVATE ENDPOINT NAME
myPEC1-ef9fd2df-df54-4dc4-9db2-978fe2d18a1b	Pending	myPEC1
myPEC2-8950f0cd-cccd-4a05-9661-53bcb4917028	Pending	myPEC2

- O administrador do servidor PostgreSQL pode optar por aprovar ou rejeitar uma PEC e, opcionalmente, adicionar uma resposta de texto curto.

Do you want to approve the connection myPEC1-ef9fd2df-df54-4dc4-9db2-978fe2d18a1b?
Request message: ""

Response message:
Approving myPEC1

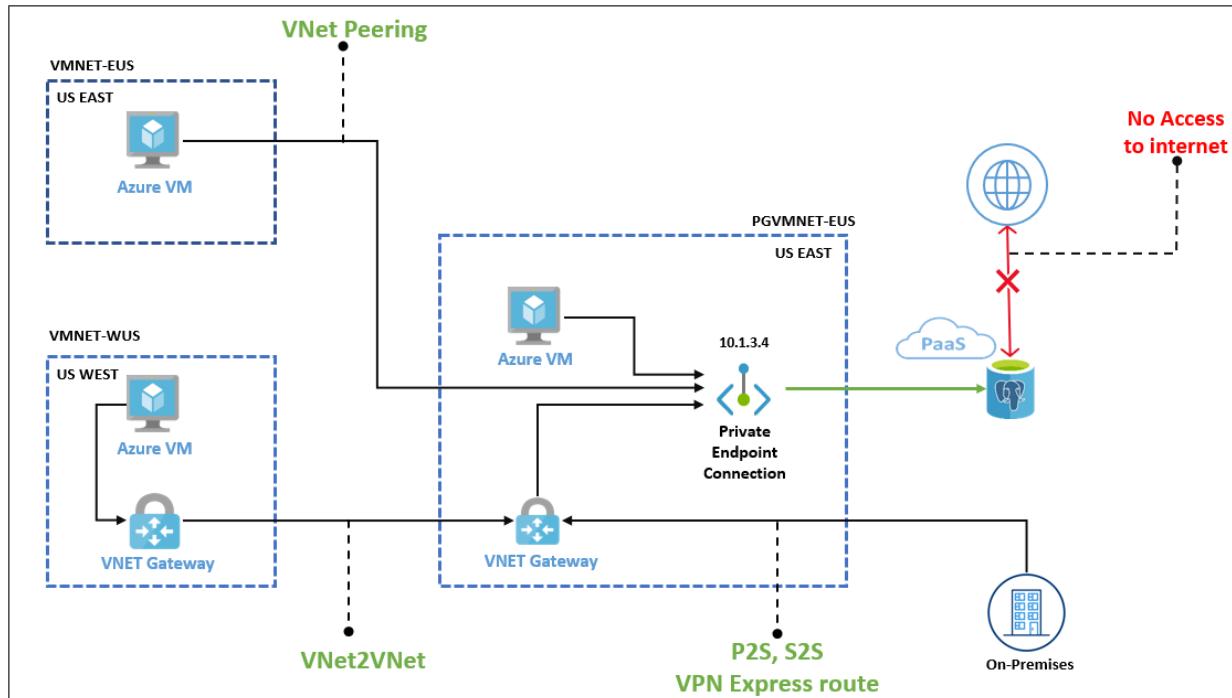
Yes No

- Após a aprovação ou a rejeição, a lista refletirá o estado apropriado junto com o texto de resposta

Connection name	State	Private endpoint name	Request/Response Message
demopostgresqlprivatelink-499c76ef-86da-44f7-bde7-da...	Approved	demopostgresqlprivatelink	Auto-approved

Casos de uso do Link Privado para o Banco de Dados do Azure para PostgreSQL

Os clientes podem se conectar ao ponto de extremidade privado a partir da mesma VNet, da [VNet emparelhada](#) na mesma região ou por meio da [conexão VNet a VNet](#) nas regiões. Além disso, os clientes podem se conectar localmente usando o ExpressRoute, o emparelhamento privado ou o túnel de VPN. Veja abaixo um diagrama simplificado que mostra os casos de uso comuns.



Como se conectar por meio de uma VM do Azure na VNET (Rede Virtual) emparelhada

Configure o [Emparelhamento VNet](#) para estabelecer a conectividade com o Banco de Dados do Azure para PostgreSQL - Servidor único por meio de uma VM do Azure em uma VNet emparelhada.

Como se conectar por meio de uma VM do Azure no ambiente VNET a VNET

Configure a [conexão de gateway de VPN VNET a VNET](#) para estabelecer a conectividade com um Banco de Dados do Azure para PostgreSQL - Servidor único por meio de uma VM do Azure em uma região ou uma assinatura diferente.

Como se conectar por meio de um ambiente local pela VPN

Para estabelecer a conectividade de um ambiente local com o banco de dados no Banco de Dados do Azure para PostgreSQL - Servidor único, escolha e implemente uma das opções:

- [Conexão ponto a site](#)
- [Conexão VPN site a site](#)
- [Círculo do ExpressRoute](#)

Link Privado combinado com regras de firewall

As seguintes situações e resultados são possíveis quando você usa o Link Privado em combinação com regras de firewall:

- Se você não configurar nenhuma regra de firewall, por padrão, nenhum tráfego será capaz de acessar o Banco de Dados do Azure para PostgreSQL - Servidor único.
- Se você configurar o tráfego público ou um ponto de extremidade de serviço e criar pontos de extremidade privados, os tipos diferentes de tráfego de entrada serão autorizados pelo tipo correspondente de regra de firewall.
- Se você não configurar nenhum tráfego público ou ponto de extremidade de serviço e criar pontos de extremidades privados, o Banco de Dados do Azure para PostgreSQL - Servidor único poderá ser

acessado somente por meio dos pontos de extremidade privados. Se você não configurar o tráfego público ou um ponto de extremidade de serviço, depois que todos os pontos de extremidades privados aprovados forem rejeitados ou excluídos, nenhum tráfego poderá acessar o Banco de Dados do Azure para PostgreSQL - Servidor único.

Negar acesso público para Banco de Dados do Azure para PostgreSQL - Servidor único

Se você quiser confiar apenas nos pontos de extremidade privados para acessar o Banco de Dados do Azure para PostgreSQL deles, poderá desabilitar a configuração de todos os pontos de extremidade públicos ([regras de firewall](#) e [pontos de extremidade de serviço de VNet](#)) definindo a configuração **Negar acesso à rede pública** no servidor e banco de dados.

Quando essa configuração é definida como *SIM*, somente as conexões por meio de pontos de extremidade privados são permitidas para o Banco de Dados do Azure para PostgreSQL. Quando essa configuração é definida como *NÃO*, os clientes podem se conectar ao Banco de Dados do Azure para PostgreSQL com base em suas configurações de firewall ou de ponto de extremidade de serviço da VNet. Além disso, quando o valor do acesso à rede privada estiver definido, os clientes não poderão adicionar e/ou atualizar as existentes "regras de firewall" e "regras de ponto de extremidade de serviço de VNet".

NOTE

Esse recurso está disponível em todas as regiões do Azure nas quais o Banco de Dados do Azure para PostgreSQL - Servidor único dá suporte aos tipos de preço "Uso Geral" e "Otimizado para Memória".

Essa configuração não tem nenhum impacto sobre as configurações de SSL e TLS para o Banco de Dados do Azure para PostgreSQL - Servidor único.

Para saber como definir a opção **Negar acesso à rede pública** para seu Banco de Dados do Azure para PostgreSQL - Servidor único do portal do Azure, consulte [Como configurar a opção de negar acesso à rede pública](#).

Próximas etapas

Para saber mais sobre os recursos de segurança do Banco de Dados do Azure para PostgreSQL - Servidor único, confira os seguintes artigos:

- Para configurar um firewall para um do Banco de Dados do Azure para PostgreSQL - Servidor único, consulte [suporte a firewall](#).
- Para saber como configurar um ponto de extremidade de serviço de rede virtual para o Banco de Dados do Azure para PostgreSQL - Servidor único, confira [Configurar o acesso de redes virtuais](#).
- Para obter uma visão geral da conectividade do Banco de Dados do Azure para PostgreSQL - Servidor único de servidor único, consulte [Arquitetura de conectividade do Banco de Dados do Azure para PostgreSQL](#)

Notificação de manutenção planejada no Banco de Dados do Azure para PostgreSQL – Servidor Único

21/05/2021 • 5 minutes to read

Saiba como se preparar para eventos de manutenção planejada no Banco de Dados do Azure para PostgreSQL.

O que é uma manutenção planejada?

O serviço de Banco de Dados do Azure para PostgreSQL executa a aplicação de patch automatizada do hardware, do sistema operacional e do mecanismo de banco de dados subjacentes. O patch inclui novas funcionalidades do serviço, segurança e atualizações de software. Para o mecanismo do PostgreSQL, as atualizações de versão secundária são automáticas e incluídas como parte do ciclo de aplicação de patch. Não há nenhuma ação do usuário nem definições de configuração necessárias para a aplicação de patch. O patch é testado extensivamente e distribuído usando as práticas de implantação seguras.

Uma manutenção planejada é uma janela de manutenção quando as atualizações de serviço são implantadas nos servidores em determinada região do Azure. Durante a manutenção planejada, um evento de notificação é criado para informar os clientes quando a atualização de serviço é implantada na região do Azure que hospeda os servidores. A duração mínima entre as duas manutenções planejadas é de 30 dias. Você receberá uma notificação da próxima janela de manutenção com 72 horas de antecedência.

Manutenção planejada – Duração e impacto sobre o cliente

Normalmente, a manutenção planejada de determinada região do Azure é completada em 15 horas. Essa janela de tempo inclui também o tempo de buffer para executar um plano de reversão, se necessário. Os servidores do banco de dados do Azure para PostgreSQL estão em execução em contêineres, de modo que as reinicializações do servidor de banco de dados normalmente levam 60-120 segundos para serem concluídas, mas não há nenhuma maneira determinística de saber quando nessa janela de 15 horas o servidor será afetado. O evento completo da manutenção planejada, incluindo as reinicializações de cada servidor, é cuidadosamente monitorado pela equipe de engenharia. O tempo de failover do servidor é dependente da recuperação do banco de dados, o que pode fazer com que o banco de dados fique on-line mais tempo se você tiver uma atividade transacional pesada no servidor no momento do failover. Para evitar um tempo de reinicialização mais longo, recomendamos evitar quaisquer transações de execução prolongada (carregamentos em massa) durante os eventos de manutenção planejada.

Em suma, enquanto o evento de manutenção planejada é executado por 15 horas, o impacto do servidor individual geralmente dura 60 segundos, dependendo da atividade transacional no servidor. Uma notificação é enviada 72 horas do calendário antes do início da manutenção planejada e outra enquanto a manutenção está em andamento em determinada região.

Como fazer para ser notificado sobre a manutenção planejada?

Você pode utilizar a funcionalidade de notificações da manutenção planejada para receber alertas para um próximo evento de manutenção planejada. Você receberá a notificação sobre as próximas 72 horas de do calendário antes de manutenção antes do evento e a outra enquanto a manutenção estiver em andamento em determinada região.

A notificação de manutenção planejada

IMPORTANT

Atualmente, as notificações de manutenção planejada estão disponíveis na pré-visualização em todas as regiões , exceto no Centro-Oeste dos EUA

As **notificações de manutenção planejada** permitem que você receba alertas para o próximo evento de manutenção planejada no Banco de Dados do Azure para PostgreSQL. Estas notificações são integradas à manutenção planejada da[Integridade do Serviço do Azure](#) e permitem que você visualiza toda a manutenção agendada para as suas assinaturas em um só local. Isto também ajuda a escalar a notificação para os públicos-alvo certos em diferentes grupos de recursos, pois você pode ter contatos diferentes responsáveis por diferentes recursos. Você receberá a notificação a próxima manutenção cerca de 72 horas do calendário antes do evento.

Faremos todas as tentativas para fornecer a **notificação de manutenção planejada** cerca de 72 horas de antecedência para todos os eventos. No entanto, em casos dos patches críticos ou de segurança, as notificações podem ser enviadas próximo ao evento ou ser omitidas.

Você pode verificar a notificação de manutenção planejada no portal do Microsoft Azure ou configurar alertas para receber notificações.

Verifique a notificação de manutenção planejada no portal do Microsoft Azure

1. No [portal do Microsoft Azure](#),selecione **Integridade do Serviço Azure**.
2. Selecione a guia **Manutenção Planejada**
3. Selecione a **Assinatura,**Região e Serviço** para os quais você deseja verificar a notificação de manutenção planejada.

Para obter a notificação de manutenção planejada

1. No [portal](#), selecione **Integridade do Serviço**.
2. Na seção **Alertas**, selecione **Alertas de integridade**.
3. Selecione + **Adicionar alerta de integridade do serviço Azure** e preencha os campos.
4. Preencha os campos requeridos.
5. Escolha o **Tipo de evento**,selecione **Manutenção planejada** ou **Selecionar tudo**
6. Em **Grupo de ações**,defina como gostaria de receber o alerta (obter um e-mail, disparar um aplicativo lógico etc.)
7. Verifique se Habilitar regra após a criação está definida como Sim.
8. Selecione **Criar regra de alerta** para concluir o alerta

Para obter as etapas detalhadas de como criar **alertas de integridade do serviço do Azure**,consulte[Criar alertas do log de atividades em notificações de serviço](#).

Posso cancelar ou adiar a manutenção planejada?

É necessária a manutenção para manter seu servidor seguro, estável e atualizado. O evento da manutenção planejada não pode ser cancelado ou adiado. Uma vez notificação ser enviada para determinada região do Azure, não é possível alterar o agendamento da aplicação de patch para um servidor individual nesta região. O patch é revertido para toda a região ao mesmo tempo. O serviço de Banco de Dados do Azure para PostgreSQL – servidor único foi projetado para aplicativos nativos de nuvem que não exigem controle granular nem personalização do serviço. Se você procura ter a capacidade de agendar a manutenção para seus servidores, recomendamos que considere o uso de [servidores flexíveis](#).

Todas as regiões do Azure são corrigidas ao mesmo tempo?

Não, todas as regiões do Azure são corrigidas durante os intervalos da janela de implantação. A janela de implantação da associação de linha geralmente se estende das 5h às 8h na hora local do dia seguinte em determinada região do Azure. As regiões do Azure de localização geográfica são corrigidas em dias diferentes. Para a alta disponibilidade e a continuidade dos negócios dos servidores de banco de dados, recomendamos aproveitar as [rélicas de leitura entre regiões](#).

Lógica de repetição

Um erro transitório, também conhecido como uma falha transitória, é um erro que será resolvido por si só.

[Erros Transitórios](#) podem ocorrer durante a manutenção. A maioria desses eventos é atenuada automaticamente pelo sistema em menos de 60 segundos. Os erros transitórios devem ser identificados por meio da [lógica de repetição](#).

Próximas etapas

- Para perguntas ou sugestões sobre como trabalhar com o Banco de Dados do Azure para PostgreSQL, envie um email para a equipe do Banco de Dados do Azure para PostgreSQL em AskAzureDBforPostgreSQL@service.microsoft.com
- Consulte [Como configurar alertas](#) para obter orientação sobre como criar um alerta em uma métrica.
- [Solucionar problemas de conexões no Banco de Dados do Azure para PostgreSQL – Servidor Único](#)
- [Tratar erros transitórios e se conectar com eficiência ao Banco de Dados do Azure para PostgreSQL – Servidor Único](#)

Visão geral da continuidade dos negócios com o Banco de Dados do Azure para PostgreSQL – Servidor Único

21/05/2021 • 6 minutes to read

Esta visão geral descreve os recursos que o Banco de Dados do Azure para PostgreSQL fornece para a continuidade dos negócios e a recuperação de desastre. Saiba mais sobre as opções para recuperação dos eventos interruptivos que podem causar perda de dados ou tornar o banco de dados e o aplicativo indisponíveis. Aprenda o que fazer quando um erro de usuário ou de aplicativo afeta a integridade dos dados, quando uma região do Azure tem uma interrupção ou quando seu aplicativo necessita de manutenção.

Recursos que podem ser utilizados para fornecer continuidade dos negócios

Na medida em que você desenvolve o plano de continuidade dos negócios, será necessário entender qual é o tempo máximo aceitável antes que o aplicativo recupere-se completamente após o evento interruptivo - esse é o RTO (Objetivo do Tempo de Recuperação). Além disso, será necessário entender a quantidade máxima de atualizações de dados recentes (intervalo de tempo) que o aplicativo poderá tolerar perder durante a recuperação após um evento interruptivo - esse é o RPO (Objetivo de Ponto de Recuperação).

O Banco de Dados do Azure para PostgreSQL fornece recursos para a continuidade dos negócios que incluem: backups com redundância geográfica com a capacidade de iniciar uma restauração geográfica, além da implantação de réplicas de leitura em uma região diferente. Cada um possui características diferentes para o tempo de recuperação e potencial perda de dados. Com o recurso de [Restauração geográfica](#), um novo servidor é criado usando os dados de backup replicados a partir de outra região. O tempo geral para restaurar e recuperar depende do tamanho dos arquivos de banco de dados e da quantidade de logs a serem recuperados. O tempo geral para estabelecer o servidor varia de alguns minutos a algumas horas. Com [réplicas de leitura](#), os logs de transações do primário são transmitidos de forma assíncrona para a réplica. No caso de uma falha do banco de dados primário devido a uma falha de nível da zona ou de nível de região, efetuar failover com a réplica fornece um RTO mais curto e uma perda de dados reduzida.

NOTE

O atraso entre o primário e a réplica depende da latência entre os sites, da quantidade de dados a serem transmitidos e, mais importante, da carga de trabalho de gravação do servidor primário. Cargas de trabalho de gravação pesadas podem gerar um atraso significativo.

Devido à natureza assíncrona da replicação usada para réplicas de leitura, elas **não devem** ser consideradas como uma solução de HA (Alta Disponibilidade), uma vez que o atraso maior pode significar RTO e OPR maiores. Somente para cargas de trabalho nas quais o atraso permanece menor por meio do horário de pico e horário fora de pico da carga de trabalho, as réplicas de leitura podem atuar como uma alternativa de HA. Caso contrário, as réplicas de leitura destinam-se a uma escala de leitura real para cargas de trabalho pesadas e para cenários de DR (Recuperação de Desastre).

A tabela a seguir compara RTO e OPR em um cenário **típico de carga de trabalho**:

RECURSO	BASIC	USO GERAL	OTIMIZADO PARA MEMÓRIA
Recuperação Pontual do backup	Qualquer ponto de restauração dentro do período de retenção RTO – Varia OPR menos de 15 min.	Qualquer ponto de restauração dentro do período de retenção RTO – Varia OPR menos de 15 min.	Qualquer ponto de restauração dentro do período de retenção RTO – Varia OPR menos de 15 min.
Restauração geográfica de backups replicados geograficamente	Sem suporte	RTO – Varia RPO < 1 h	RTO – Varia RPO < 1 h
Rélicas de leitura	RTO – Minutos* OPR menos de 5 min.*	RTO – Minutos* OPR menos de 5 min.*	RTO – Minutos* OPR menos de 5 min.*

* O RTO e o OPR **podem ser muito maiores** em alguns casos, dependendo de vários fatores, incluindo a latência entre sites, a quantidade de dados a serem transmitidos e, principalmente, a carga de trabalho de gravação do banco de dados primário.

Recuperar um servidor após um erro de aplicativo ou usuário

Você pode usar os backups do serviço para recuperar um servidor de vários eventos interruptivos. Um usuário pode excluir alguns dados acidentalmente, remover uma tabela importante inadvertidamente ou até mesmo um banco de dados inteiro. Um aplicativo pode substituir acidentalmente dados corretos por dados incorretos, devido a uma falha de aplicativo, e assim por diante.

Você pode executar uma **restauração pontual** para criar uma cópia do servidor em um ponto no tempo conhecido e ideal. Esse ponto no tempo deve estar dentro do período de retenção de backup que você configurou para o servidor. Depois que os dados forem restaurados para o novo servidor, você poderá substituir o servidor de origem pelo servidor restaurado recentemente, ou copiar os dados necessários do servidor restaurado para o servidor de origem.

IMPORTANT

Excluir servidores **não é possível** ser restaurado. Se você excluir o servidor, todos os bancos de dados que pertencem ao servidor também serão excluídos e não poderão ser recuperados. Use o [bloqueio de recursos do Azure](#) para ajudar a evitar a exclusão acidental do seu servidor.

Recuperar de uma interrupção no data center do Azure

Embora seja raro, um data center do Azure pode ter uma interrupção. Quando uma interrupção ocorre, isso causa uma interrupção dos negócios, que pode durar alguns minutos ou horas.

Uma opção é aguardar até que o servidor retorne online quando a interrupção do data center terminar. Isso funciona para aplicativos que podem ter o servidor offline por algum período de tempo, por exemplo, um ambiente de desenvolvimento. Quando o data center tem uma interrupção, não é possível saber por quanto tempo a interrupção poderá durar, então, essa opção somente funcionará se você não precisar usar o servidor por algum tempo.

Restauração geográfica

O recurso de restauração geográfica restaura o servidor usando backups com redundância geográfica. Os backups são hospedados na [região emparelhada](#) do servidor. É possível restaurar a partir desses backups para qualquer outra região. A restauração geográfica cria um novo servidor com os dados dos backups. Saiba mais

sobre a restauração geográfica no [artigo de conceitos de backup e restauração](#).

IMPORTANT

A restauração geográfica somente será possível se o servidor foi provisionado com armazenamento de backup com redundância geográfica. Se você deseja alternar de backups redundantes localmente para redundantes geo-ativos para um servidor existente, você deve fazer um dump usando o pg_dump do seu servidor existente e restaurá-lo em um servidor recém-criado configurado com backups geo-redundantes.

Réplicas de leitura entre regiões

Você pode usar réplicas de leitura entre regiões para aprimorar sua continuidade de negócios e planejamento de recuperação de desastre. As réplicas de leitura são atualizadas assincronamente usando a tecnologia de replicação física do PostgreSQL e podem atrasar o primário. Saiba mais sobre réplicas de leitura, regiões disponíveis e como fazer failover no [artigo de conceitos de réplicas de leitura](#).

Perguntas frequentes

Onde o Banco de Dados do Azure para PostgreSQL armazena dados do cliente?

Por padrão, o Banco de Dados do Azure para PostgreSQL não move nem armazena dados do cliente fora da região na qual está implantado. No entanto, os clientes podem optar por habilitar [backups com redundância geográfica](#) ou criar [réplica de leitura entre regiões](#) para armazenar dados em outra região.

Próximas etapas

- Para saber mais sobre backups automáticos, confira [Backups automáticos no Banco de Dados do Azure para PostgreSQL](#).
- Saiba como restaurar usando o [portal do Azure](#) ou a [CLI do Azure](#).
- Saiba mais sobre [réplicas de leitura no Banco de Dados do Azure para PostgreSQL](#).

Alta disponibilidade em Servidor único do Banco de Dados do Azure para PostgreSQL

21/05/2021 • 7 minutes to read

O serviço de Servidor único do Banco de Dados do Azure para PostgreSQL fornece um alto nível de disponibilidade garantido com o SLA (contrato de nível de serviço) com suporte financeiro de **99,99%** de tempo de atividade. O Banco de Dados do Azure para PostgreSQL fornece alta disponibilidade durante eventos planejados, como a operação de computação dimensionada iniciada pelo usuário, e também quando ocorrem eventos não planejados, como falhas de rede de hardware ou software subjacentes. O Banco de Dados do Azure para PostgreSQL pode se recuperar rapidamente das circunstâncias mais críticas, garantindo praticamente nenhum tempo de inatividade do aplicativo ao usar esse serviço.

O BD do Azure para PostgreSQL é adequado para a execução de dados de missão crítica que exigem alto tempo de atividade. Baseado na arquitetura do Azure, o serviço tem recursos inerentes de alta disponibilidade, redundância e resiliência para reduzir o tempo de inatividade do banco de dados de interrupções planejadas e não planejadas, sem a necessidade de configurar outro componente adicional.

Componentes no Servidor único do Banco de Dados do Azure para PostgreSQL

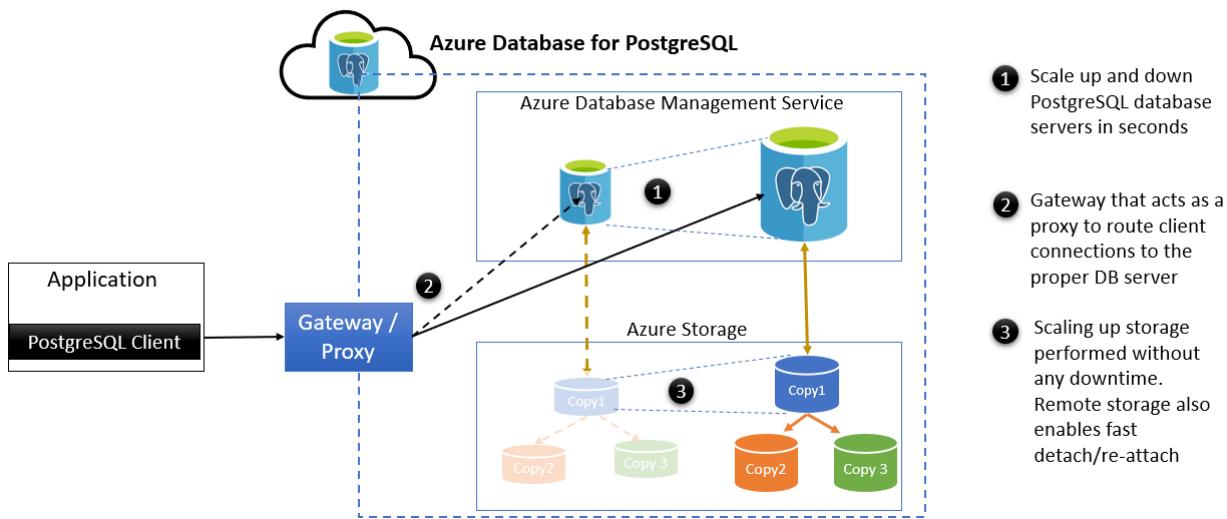
COMPONENTE	DESCRIÇÃO
Servidor de Banco de Dados PostgreSQL	O Banco de Dados do Azure para PostgreSQL fornece segurança, isolamento, proteções de recursos e capacidade de reinicialização rápida para servidores de banco de dados. Esses recursos facilitam operações como o dimensionamento e a operação de recuperação do servidor de banco de dados após ocorrer uma interrupção em segundos. Normalmente, as modificações nos servidores de dados ocorrem durante uma transação de um banco de dados. Todas as alterações de banco de dados são gravadas de forma síncrona na maneira de WAL (logs write-ahead) no Armazenamento do Microsoft Azure – que é anexado ao servidor de banco de dados. Durante o processo de ponto de verificação do banco de dados, as páginas de dados da memória do servidor de banco de dados também são liberadas para o armazenamento.
Armazenamento Remoto	Todos os arquivos de dados físicos do PostgreSQL e os arquivos WAL são armazenados no Armazenamento do Microsoft Azure, que é arquitetado para armazenar três cópias de dados em uma região para garantir a redundância, a disponibilidade e a confiabilidade dos dados. A camada de armazenamento também é independente do servidor de banco de dados. Ela pode ser desanexada de um servidor de banco de dados com falha e reanexada em um novo servidor de banco de dados em questão de segundos. Além disso, o Armazenamento do Microsoft Azure monitora continuamente as falhas de armazenamento. Se uma corrupção de bloco for detectada, ela será automaticamente corrigida instanciando uma nova cópia de armazenamento.

COMPONENTE	DESCRIÇÃO
Gateway	O gateway atua como um proxy de banco de dados roteando todas as conexões de cliente para o servidor de banco de dados.

Mitigação de tempo de inatividade planejada

O Banco de Dados do Azure para PostgreSQL é projetado para fornecer alta disponibilidade durante operações de tempo de inatividade planejadas.

Elastic Scaling with Built-in High Availability



1. Aumentar e reduzir os servidores de banco de dados do PostgreSQL em segundos
2. O gateway que atua como um proxy para rotear o cliente se conecta ao servidor de banco de dados apropriado
3. O aumento do armazenamento pode ser realizado sem nenhum tempo de inatividade. O armazenamento remoto permite desanexar/reanexar rapidamente após o failover. Estes são alguns cenários de manutenção planejada:

CENÁRIO	DESCRIÇÃO
Dimensionar/reduzir verticalmente a computação	Quando o usuário executa a operação de dimensionar/reduzir verticalmente, um novo servidor de banco de dados é provisionado usando a configuração de computação dimensionada. No servidor de banco de dados antigo, os pontos de verificação ativos têm permissão para serem concluídos, as conexões de cliente são descarregadas, todas as transações não confirmadas são canceladas e desligadas. Então o armazenamento é desanexado do servidor de banco de dados antigo e anexado ao servidor de banco de dados novo. Quando o aplicativo cliente tenta se reconectar ou tenta fazer uma nova conexão, o gateway direciona a solicitação de conexão para o servidor de banco de dados novo.
Dimensionamento do armazenamento	Dimensionar o armazenamento é uma operação online e não interrompe o servidor de banco de dados.

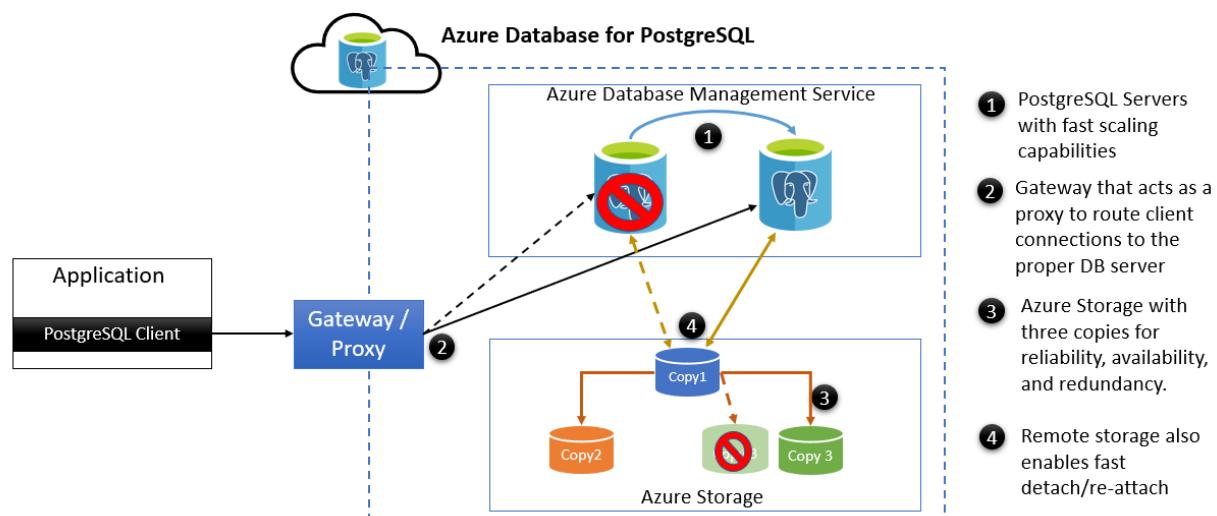
CENÁRIO	DESCRIÇÃO
Implantação de software novo (Azure)	Novos recursos de distribuição ou correções de bugs ocorrem automaticamente como parte da manutenção planejada do serviço. Para saber mais, consulte a documentação e também verifique o portal .
Atualizações secundárias de versão	O Banco de Dados do Azure para PostgreSQL corrige automaticamente os servidores de banco de dados para a versão secundária determinada pelo Azure. Isso acontece como parte da manutenção planejada do serviço. Isso causará em alguns segundos de tempo de inatividade e o servidor de banco de dados será reiniciado automaticamente com a nova versão secundária. Para saber mais, consulte a documentação e também verifique o portal .

Mitigação de tempo de inatividade não planejada

O tempo de inatividade não planejado pode ocorrer como resultado de falhas imprevistas, incluindo falhas de hardware subjacentes, problemas de rede e bugs de software. Se o servidor de banco de dados ficar inativo inesperadamente, um novo servidor de banco de dados será provisionado automaticamente em segundos. O armazenamento remoto é anexado automaticamente ao novo servidor de banco de dados. O mecanismo do PostgreSQL executa a operação de recuperação usando WAL e arquivos de banco de dados abrindo o servidor de banco de dados para permitir que os clientes se conectem. As transações não confirmadas são perdidas e precisam ser repetidas pelo aplicativo. Embora não seja possível evitar um tempo de inatividade não planejado, o Banco de Dados do Azure para PostgreSQL reduz o tempo de inatividade realizando automaticamente operações de recuperação no servidor de banco de dados e nas camadas de armazenamento sem a necessidade de intervenção humana.

Built-in High Availability

99.99% Uptime SLA



1. Servidores PostgreSQL do Azure com recursos de dimensionamento rápido.
2. O gateway que atua como um proxy para rotear as conexões do cliente para o servidor de banco de dados apropriado
3. Armazenamento do Azure com três cópias para confiabilidade, disponibilidade e redundância.
4. O armazenamento remoto permite desanexar/reanexar rapidamente após o failover do servidor.

Tempo de inatividade não planejado: cenários de falha e recuperação de serviço

Aqui estão alguns cenários de falha e como o Banco de Dados do Azure para PostgreSQL se recupera

automaticamente:

CENÁRIO	RECUPERAÇÃO AUTOMÁTICA
Falha no servidor de banco de dados	<p>Se o servidor de banco de dados estiver inoperante devido a alguma falha de hardware subjacente, as conexões ativas serão removidas e todas as transações em andamento serão anuladas. Um novo servidor de banco de dados é implantado automaticamente, e o armazenamento remoto é anexado ao novo servidor de banco de dado. Depois que a recuperação do banco de dados for concluída, os clientes podem se conectar ao novo servidor de banco de dados por meio do gateway.</p> <p>O RTO (tempo de recuperação) depende de vários fatores, incluindo a atividade no momento da falha, como transação grande e a quantidade de recuperação a ser executada durante o processo de inicialização do servidor de banco de dados.</p> <p>Os aplicativos que usam os bancos de dados do PostgreSQL precisam ser criados para detectar e repetir as conexões e transações que falharam. Quando o aplicativo repete a operação, o gateway redireciona a conexão de forma transparente para o servidor de banco de dados recém-criado.</p>
Falha de armazenamento	<p>Os aplicativos não detectam o impacto de problemas relacionados ao armazenamento, como uma falha de disco ou um corrompimento de bloco físico. Como os dados são armazenados em 3 cópias, a cópia dos dados é atendida pelo armazenamento remanescente. As corrupções de bloco são corrigidas automaticamente. Se uma cópia dos dados for perdida, uma nova cópia dos dados será criada automaticamente.</p>

Aqui estão alguns cenários de falha que necessitam da ação do usuário para serem recuperados:

CENÁRIO	PLANO DE RECUPERAÇÃO
Falha de região	<p>A falha de uma região é um evento raro. No entanto, se precisar de proteção de uma falha de região, você poderá configurar uma ou mais réplicas de leitura em outras regiões para DR (recuperação de desastre). (Consulte este artigo, sobre como criar e gerenciar réplicas de leitura, para mais detalhes). No caso de uma falha em nível de região, você pode promover manualmente a réplica de leitura configurada na outra região para ser seu servidor de banco de dados de produção.</p>

CENÁRIO	PLANO DE RECUPERAÇÃO
Erros de usuário/lógicos	<p>A recuperação de erros do usuário, como tabelas removidas por acidente ou dados atualizados incorretamente, envolve a execução de uma PITR (recuperação pontual), que restaura e recupera os dados até o momento anterior da ocorrência do erro.</p> <p>Se você quiser restaurar apenas um subconjunto de bancos de dados ou tabelas específicas, em vez de todos os bancos de dados no servidor de banco de dados, poderá restaurar o servidor de banco de dados em uma nova instância, exportar as tabelas por meio de <code>pg_dump</code> e usar <code>pg_restore</code> para restaurar essas tabelas em seu banco de dados.</p>

Resumo

O Banco de Dados do Azure para PostgreSQL fornece recursos para reiniciar rapidamente servidores de banco de dados, armazenamento redundante e roteamento eficiente do gateway. Para proteção de dados adicional, você pode configurar os backups para serem replicados geograficamente e também implantar uma ou mais réplicas de leitura em outras regiões. Com os recursos inerentes de alta disponibilidade, o Banco de Dados do Azure para PostgreSQL protege seus bancos de dados contra as interrupções mais comuns e oferece um SLA de tempo de atividade, com suporte financeiro potencial do setor, de [99,99%](#). Todos esses recursos de disponibilidade e confiabilidade permitem que o Azure seja a plataforma ideal para executar aplicativos críticos.

Próximas etapas

- Saiba mais sobre [Regiões do Azure](#)
- Saiba mais sobre [tratamento de erros de conectividade transitórios](#)
- Saiba como [replicar seus dados com réplicas de leitura](#)

Backup e restauração no Banco de Dados do Azure para PostgreSQL – Servidor Único

09/08/2021 • 9 minutes to read

O Banco de Dados do Azure para PostgreSQL cria backups de servidor automaticamente e os armazena no armazenamento com redundância geográfica ou local configurado pelo usuário. Os backups podem ser usados para restaurar o servidor pontualmente. Os recursos de backup e restauração são uma parte essencial de qualquer estratégia de continuidade dos negócios, pois eles protegem seus dados contra exclusão ou corrupção acidentais.

Backups

O Banco de Dados do Azure para PostgreSQL faz backups dos arquivos de dados e do log de transações. Dependendo do tamanho máximo de armazenamento com suporte, fazemos backups totais e diferenciais (servidores de armazenamento máximo de 4 TB) ou backups de instantâneo (servidores de armazenamento máximo de até 16 TB). Esses backups permitem que você restaure um servidor pontualmente dentro de seu período de retenção de backup configurado. O período de retenção de backup padrão é de sete dias. Você pode, opcionalmente, configurá-lo para até 35 dias. Todos os backups são criptografados usando a criptografia AES de 256 bits.

Esses arquivos de backup não podem ser exportados. Os backups podem ser usados somente para operações de restauração no Banco de Dados do Azure para PostgreSQL. Você pode utilizar [pg_dump](#) para copiar um banco de dados.

Frequência de backup

Servidores com armazenamento de até 4 TB

Para servidores que dão suporte a até 4 TB de armazenamento máximo, os backups completos ocorrem uma vez a cada semana. Os backups diferenciais ocorrem duas vezes ao dia. Os backups de log de transações ocorrem a cada cinco minutos.

Servidores com armazenamento de até 16 TB

Em um subconjunto de [regiões do Azure](#), todos os servidores recentemente provisionados podem dar suporte ao armazenamento de uso geral de até 16 TB. Os backups nos servidores de armazenamento de 16 TB são baseados em instantâneos. O primeiro backup de instantâneo completo é agendado imediatamente após a criação de um servidor. O primeiro backup de instantâneo completo é mantido como o backup de base do servidor. Os backups de instantâneo subsequentes são apenas backups diferenciais. Os backups de instantâneo diferenciais não ocorrem em um agendamento fixo. Em um dia, três backups de instantâneo diferenciais são executados. Os backups de log de transações ocorrem a cada cinco minutos.

NOTE

Os backups automáticos são executados para [servidores de réplica](#) que são configurados com até 4 TB de configuração de armazenamento.

Retenção de backup

Os backups são mantidos no servidor com base na configuração do período de retenção de backup. É possível definir um período de retenção de 7 a 35 dias. O período de retenção padrão é de 7 dias. É possível definir o período de retenção durante a criação do servidor ou posteriormente, atualizando a configuração de backup pelo [portal do Azure](#) ou pela [CLI do Azure](#).

O período de retenção de backup determina até quando a restauração de pontos anteriores pode ser feita, já que ele se baseia em backups disponíveis. O período de retenção de backup também pode ser tratado como uma janela de recuperação sob uma perspectiva de restauração. Todos os backups necessários para executar uma restauração pontual dentro do período de retenção de backup são mantidos no armazenamento de backup. Por exemplo, se o período de retenção de backup for definido como 7 dias, a janela de recuperação será considerada nos últimos 7 dias. Nesse cenário, são mantidos todos os backups necessários para restaurar o servidor nos últimos 7 dias. Com uma janela de retenção de backup de sete dias:

- Os servidores com armazenamento de até 4 TB manterão até dois backups de banco de dados completos, todos os backups diferenciais e backups de log de transações executados desde o backup de banco de dados completo mais antigo.
- Os servidores com armazenamento de até 16 TB manterão o instantâneo completo do banco de dados, todos os instantâneos diferenciais e backups de log de transações dos últimos 8 dias.

Opções de redundância de backup

O Banco de Dados do Azure para PostgreSQL fornece a flexibilidade de escolher entre o armazenamento de backup com redundância local ou com redundância geográfica nas camadas de Uso Geral e Otimizado para Memória. Quando os backups são armazenados no armazenamento de backup com redundância geográfica, eles não são somente armazenados dentro da região em que o servidor está hospedado, mas também replicados em um [datacenter emparelhado](#). Isso fornece maior proteção e capacidade de restaurar o servidor em uma região diferente em caso de desastre. A camada Básica oferece apenas o armazenamento de backup de redundância local.

IMPORTANT

A configuração do armazenamento com redundância local ou geográfica para backup só é permitida durante a criação do servidor. Quando o servidor é provisionado, você não pode alterar a opção de redundância do armazenamento de backup.

Custo do armazenamento de backup

O Banco de Dados do Azure para PostgreSQL fornece até 100% de seu armazenamento de servidor configurado como armazenamento de backup sem custo adicional. Qualquer armazenamento de backup adicional usado será cobrado em GB por mês. Por exemplo, caso tenha provisionado um servidor com 250 GB de armazenamento, terá 250 GB de armazenamento adicional disponíveis para backups de servidor, sem nenhum custo adicional. O armazenamento consumido para backups com mais de 250 GB é cobrado de acordo com o [modelo de preços](#).

É possível usar a métrica [Armazenamento de Backup usado](#) no Azure Monitor, disponível no portal do Azure, para monitorar o armazenamento de backup consumido por um servidor. A métrica Armazenamento de Backup usado representa a soma do armazenamento consumido por todos os backups de banco de dados, backups diferenciais e backups de log retidos com base no período de retenção de backup definido para o servidor. A frequência dos backups é gerenciada pelo serviço e foi explicada anteriormente. Uma atividade transacional intensa no servidor pode fazer com que o uso do armazenamento de backup aumente, independentemente do tamanho total do banco de dados. Para o armazenamento com redundância geográfica, o uso de armazenamento de backup é o dobro do armazenamento com redundância local.

O principal meio de controlar o custo de armazenamento de backup é definir o período de retenção de backup apropriado e escolher as opções de redundância de backup corretas para atender aos objetivos de recuperação que você deseja. É possível definir um período de retenção no intervalo de 7 a 35 dias. Para servidores de uso geral e otimizados de memória existe a opção de armazenamento com redundância geográfica para backups.

Restaurar

No Banco de Dados do Azure para PostgreSQL, a execução de uma restauração cria um novo servidor de backup do servidor original.

Há dois tipos de restauração disponíveis:

- A **Restauração pontual** está disponível em qualquer opção de redundância de backup e cria um novo servidor na mesma região do servidor original.
- A **Restauração geográfica** está disponível somente se você configurou seu servidor para armazenamento com redundância geográfica; ele permite que você restaure o servidor em uma região diferente.

O tempo estimado de recuperação dependerá de vários fatores, incluindo os tamanhos dos bancos de dados, o tamanho do log de transações, a largura de banda de rede e o número total de bancos de dados de recuperação na mesma região e ao mesmo tempo. O tempo de recuperação varia dependendo do último backup de dados e da quantidade de recuperação que precisa ser executada. Geralmente é menor do que 12 horas.

NOTE

Se o servidor PostgreSQL de origem estiver criptografado com chaves gerenciadas pelo cliente, veja a [documentação](#) para obter considerações adicionais.

NOTE

Se você quiser restaurar um servidor PostgreSQL excluído, siga o procedimento documentado [aqui](#).

Restauração em um momento determinado

Independentemente de sua opção de redundância de backup, você pode executar uma restauração para qualquer ponto anterior dentro de seu período de retenção de backup. Um novo servidor é criado na mesma região do Azure do servidor original. Ele é criado com a configuração do servidor original para o tipo de preço, a geração de computação, o número de núcleos virtuais, o tamanho do armazenamento, o período de retenção de backup e a opção de redundância de backup.

A Restauração pontual é útil em vários cenários. Por exemplo, quando um usuário exclui dados acidentalmente, descarta uma tabela ou um banco de dados importante, ou se um aplicativo acidentalmente substitui dados bons por dados inválidos devido a um defeito no aplicativo.

Talvez seja necessário aguardar a execução do próximo backup de log de transações antes de poder restaurar para um ponto anterior nos últimos cinco minutos.

Se você quiser restaurar uma tabela descartada,

1. Restaure o servidor de origem usando o método pontual.
2. Faça uma cópia de backup da tabela usando `pg_dump` do servidor restaurado.
3. Renomeie a tabela de origem no servidor original.
4. Importe a tabela usando a linha de comando `psql` no servidor original.
5. Opcionalmente, você pode excluir o servidor restaurado.

NOTE

É recomendável não criar várias restaurações para o mesmo servidor ao mesmo tempo.

Restauração geográfica

É possível restaurar um servidor para outra região do Azure onde o serviço está disponível caso você tenha configurado o servidor para backups com redundância geográfica. Os servidores que dão suporte a até 4 TB de

armazenamento podem ser restaurados para a região emparelhada geograficamente ou para qualquer região que ofereça suporte a até 16 TB de armazenamento. Para servidores que dão suporte a até 16 TB de armazenamento, os backups geográficos podem ser restaurados em qualquer região que ofereça suporte a servidores de 16 TB também. Examine os [tipos de preço do Banco de Dados do Azure para PostgreSQL](#) para ver a lista de regiões com suporte.

A restauração geográfica é a opção de recuperação padrão quando o servidor não está disponível devido a um incidente na região em que ele está hospedado. Se um incidente de grande escala em uma região resultar na indisponibilidade do seu aplicativo de banco de dados, você poderá restaurar um servidor do backup com redundância geográfica para um servidor em qualquer outra região. Há um atraso entre quando um backup é feito e quando ele é replicado em uma região diferente. Esse atraso pode ser de até uma hora, então, em caso de desastre pode haver perda de dados de até uma hora.

Durante a restauração geográfica, as configurações de servidor que podem ser alteradas incluem as opções de geração de computação, vCore, período de retenção de backup e redundância de backup. Não há suporte para alterar o tipo de preço (básico, uso geral ou com otimização de memória) ou tamanho de armazenamento.

NOTE

Se o servidor de origem usar a criptografia dupla de infraestrutura, para restaurar o servidor, haverá limitações, incluindo regiões disponíveis. Veja a [infraestrutura de criptografia dupla](#) para obter mais detalhes.

Executar tarefas de pós-restauração

Após uma restauração de um dos mecanismos de recuperação, você deve executar as seguintes tarefas para colocar os usuários e os aplicativos novamente em execução:

- Se o novo servidor for usado para substituir o servidor original, redirecione clientes e aplicativos de cliente para o novo servidor. Além disso, altere também o nome de usuário para `username@new-restored-server-name`.
- Verifique se as regras de VNet e de firewall no nível do servidor adequadas estão em vigor para que os usuários se conectem. Essas regras não são copiadas do servidor original.
- Verifique se as permissões e os logons adequados no nível do banco de dados estão em vigor
- Configurar os alertas, conforme apropriado

Próximas etapas

- Saiba como restaurar usando o[portal do Azure](#).
- Saiba como restaurar usando a[CLI do Azure](#).
- Para saber mais sobre continuidade dos negócios, confira a [visão geral de continuidade dos negócios](#).

Monitorar e ajustar o Banco de Dados do Azure para PostgreSQL – Servidor único

25/05/2021 • 4 minutes to read

Monitorar os dados dos seus servidores ajuda a solucionar problemas e otimizar sua carga de trabalho. O Banco de Dados do Azure para PostgreSQL oferece várias opções de monitoramento para fornecer insights sobre o comportamento do servidor.

Métricas

O Banco de Dados do Azure para PostgreSQL oferece várias métricas que fornecem informações sobre o comportamento dos recursos compatíveis com o servidor PostgreSQL. Cada métrica é emitida em uma frequência de um minuto e tem até [93 dias de histórico](#). É possível configurar alertas nas métricas. Para obter diretrizes passo a passo, consulte [How to set up alerts](#) (Como configurar alertas). Outras tarefas incluem a configuração de ações automatizadas, execução de análises avançadas e arquivamento de histórico. Para obter mais informações, consulte a [Visão geral das métricas no Microsoft Azure](#).

Listar métricas

Essas métricas estão disponíveis para o Banco de Dados do Azure para PostgreSQL:

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRIÇÃO
cpu_percent	Porcentagem de CPU	Porcentagem	O percentual de CPU em uso.
memory_percent	Porcentagem de memória	Porcentagem	O percentual de memória em uso.
io_consumption_percent	Porcentagem de E/S	Porcentagem	O percentual de E/S em uso. (Não aplicável para servidores de camada Básica.)
storage_percent	Porcentagem de armazenamento	Porcentagem	O percentual de armazenamento usado fora do máximo do servidor.
storage_used	Armazenamento usado	Bytes	A quantidade de armazenamento em uso. O armazenamento usado pelo serviço pode incluir os arquivos de banco de dados, os logs de transação e os logs do servidor.
storage_limit	Límite de armazenamento	Bytes	O armazenamento máximo para esse servidor.

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRIÇÃO
serverlog_storage_percent	Porcentagem de armazenamento do Log do Servidor	Porcentagem	A porcentagem de armazenamento de log do servidor usada fora do armazenamento de log máximo do servidor.
serverlog_storage_usage	Armazenamento do Log do Servidor usado	Bytes	A quantidade de armazenamento de log do servidor em uso.
serverlog_storage_limit	Limite de armazenamento do Log do Servidor	Bytes	O armazenamento de log do servidor de máximo para esse servidor.
active_connections	Conexões ativas	Contagem	O número de conexões ativas com o servidor.
connections_failed	Conexões com falha	Contagem	O número de conexões estabelecidas que falharam.
network_bytes_egress	Saída da rede	Bytes	Rede-Out em conexões ativas.
network_bytes_ingress	Entrada na rede	Bytes	Entrada de rede em conexões ativas.
backup_storage_used	Backup do Microsoft Azure	Bytes	A quantidade de armazenamento de backup usado. A métrica representa a soma do armazenamento consumido por todos os backups de banco de dados, backups diferenciais e backups de log retidos com base no período de retenção de backup definido para o servidor. A frequência dos backups é gerenciada pelo serviço e explicada no artigo de conceitos . Para o armazenamento com redundância geográfica, o uso de armazenamento de backup é o dobro do armazenamento com redundância local.
pg_replica_log_delay_in_bytes	Retardo Máximo entre Réplicas	Bytes	O retardo em bytes entre a réplica primária e a réplica com o maior retardo. Essa métrica está disponível apenas no servidor primário.

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRIÇÃO
pg_replica_log_delay_in_seconds	Retardo da Réplica	Segundos	O tempo desde a última transação reproduzida. Essa métrica está disponível apenas para servidores de réplica.

Logs do servidor

Você pode habilitar o registro no servidor. Esses logs de recursos podem ser enviados para [logs do Azure Monitor](#), Hubs de Eventos e uma Conta de Armazenamento. Para saber mais sobre o registro em log, visite a página [logs de servidor](#).

Repositório de Consultas

O [Repositório de Consultas](#) mantém o controle do desempenho da consulta ao longo do tempo, incluindo eventos de espera e estatísticas de runtime de consulta. O recurso mantém as informações de desempenho de runtime de consulta em um banco de dados do sistema chamado `azure_sys` sob o esquema `query_store`. Você pode controlar a coleta e o armazenamento de dados por meio de vários botões de configuração.

Análise de Desempenho de Consultas

[Análise de Desempenho de Consultas](#) funciona em conjunto com o Repositório de Consultas para fornecer visualizações acessíveis do portal do Azure. Esses gráficos permitem que você identifique as principais consultas que afetam o desempenho. A Análise de Desempenho de Consultas está acessível na seção **Supporte + solução de problemas** da página do portal do servidor do Banco de Dados do Azure para PostgreSQL.

Recomendações de desempenho

O recurso [Recomendações de Desempenho](#) identifica as oportunidades de melhorar o desempenho da carga de trabalho. As Recomendações de Desempenho fornecem recomendações para a criação de novos índices que têm o potencial de melhorar o desempenho de suas cargas de trabalho. Para produzir recomendações de índice, o recurso leva em consideração várias características do banco de dados, inclusive seu esquema e a carga de trabalho, conforme relatado pelo Repositório de Consultas. Depois de implementar qualquer recomendação de desempenho, os clientes devem testar o desempenho para avaliar o impacto dessas alterações.

Notificação de manutenção planejada

As [notificações de manutenção planejada](#) permitem que você receba alertas para a próxima manutenção planejada no Banco de Dados do Azure para PostgreSQL – Servidor único. Estas notificações são integradas à manutenção planejada da [Integridade do Serviço do Azure](#) e permitem que você visualize toda a manutenção agendada para as suas assinaturas em um só local. Isto também ajuda a escalar a notificação para os públicos-alvo certos em diferentes grupos de recursos, pois você pode ter contatos diferentes responsáveis por diferentes recursos. Você receberá a notificação sobre a próxima manutenção 72 horas antes do evento.

Saiba mais sobre como configurar notificações no documento [notificações de manutenção planejada](#).

Próximas etapas

- Confira [como configurar alertas](#) para obter orientação sobre como criar um alerta em uma métrica.
- Para obter mais informações sobre como acessar e exportar métricas usando o portal do Microsoft Azure, a API REST ou a CLI, veja a [Visão geral das métricas do Azure](#)

- Leia nosso blog sobre [práticas recomendadas para monitorar seu servidor](#).
- Saiba mais sobre as [notificações de manutenção planejada](#) no Banco de Dados do Azure para PostgreSQL – Servidor único.

Logs no Banco de Dados do Azure para PostgreSQL – Servidor Único

09/08/2021 • 5 minutes to read

O Banco de Dados do Azure para PostgreSQL permite configurar e acessar os logs padrão do Postgres. Esses logs podem ser usados para identificar, solucionar problemas e reparar erros de configuração e desempenho abaixo do ideal. As informações de registro em log que você pode configurar e acessar incluem erros, informações de consulta, registros de vácuo automático, conexões e pontos de verificação. (O acesso aos logs de transação não está disponível).

O log de auditoria é disponibilizado por meio de uma extensão Postgres, pgaudit. Para saber mais, acesse o artigo [conceitos de auditoria](#).

Configurar o registro em log

Você pode configurar o log padrão do Postgres no seu servidor usando os parâmetros de registro em log. Em cada servidor de Banco de Dados do Azure para PostgreSQL, `log_checkpoints` e `log_connections` estão ativados por padrão. Há parâmetros adicionais que você pode ajustar para atender às suas necessidades de registro em log:

Server Parameters		
Allows changing of PostgreSQL server parameters		
Save Discard		
<input type="text"/> Search to filter items...		
PARAMETER NAME	VALUE	DESCRIPTION
<code>client_min_messages</code>	<input type="button" value="NOTICE"/> <input type="button" value="OFF"/>	Sets the message levels that are sent to the client.
<code>debug_print_parse</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs each query's parse tree.
<code>debug_print_plan</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs each query's execution plan.
<code>debug_print_rewritten</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs each query's rewritten parse tree.
<code>log_checkpoints</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs each checkpoint.
<code>log_connections</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs each successful connection.
<code>log_disconnections</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs end of a session, including duration.
<code>log_duration</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs the duration of each completed SQL statement.
<code>log_error_verbosity</code>	<input type="button" value="DEFAULT"/> <input type="button" value=""/>	Sets the verbosity of logged messages.
<code>log_lock_waits</code>	<input type="button" value="ON"/> <input type="button" value="OFF"/>	Logs long lock waits.
<code>log_min_duration_statement</code>	<input type="text" value="-1"/> <input type="button" value=""/>	Sets the minimum execution time (in milliseconds) above which statements will be logged. -
<code>log_min_error_statement</code>	<input type="button" value="ERROR"/> <input type="button" value=""/>	Causes all statements generating error at or above this level to be logged.
<code>log_min_messages</code>	<input type="button" value="WARNING"/> <input type="button" value=""/>	Sets the message levels that are logged.
<code>log_retention_days</code>	<input type="text" value="3"/> <input type="button" value=""/>	Sets how many days a log file is saved for.
<code>log_statement</code>	<input type="button" value="NONE"/> <input type="button" value=""/>	Sets the type of statements logged.

Para saber mais sobre os parâmetros de log do Postgres, visite as seções [Quando fazer um registro em log](#) e [O que registrar em log](#) na documentação do Postgres. A maioria, mas não todos, dos parâmetros de log do Postgres, estão disponíveis para configurar no Banco de Dados do Azure para PostgreSQL.

Para saber como configurar parâmetros no Banco de Dados do Azure para PostgreSQL, consulte a [documentação do portal](#) ou a [documentação da CLI](#).

NOTE

Configurar um alto volume de logs, por exemplo, registro de instruções em log, pode adicionar uma sobrecarga de desempenho significativa.

Acessar arquivos de .log

O formato de log padrão no Banco de Dados do Azure para PostgreSQL é .log. Uma linha de exemplo desse log tem esta aparência:

```
2019-10-14 17:00:03 UTC-5d773cc3.3c-LOG: connection received: host=101.0.0.6 port=34331 pid=16216
```

O Banco de Dados do Azure para PostgreSQL fornece um local de armazenamento de curto prazo para os arquivos .log. Um novo arquivo começa a cada 1 hora ou 100 MB, o que ocorrer primeiro. Os logs são anexados ao arquivo atual à medida que são emitidos do Postgres.

Você pode definir o período de retenção desse armazenamento de logs de curto prazo usando o parâmetro `log_retention_period`. O valor padrão é de 3 dias; o valor máximo é de 7 dias. O local de armazenamento de curto prazo pode conter até 1 GB de arquivos de log. Após 1 GB, os arquivos mais antigos, independentemente do período de retenção, serão excluídos para liberar espaço para novos logs.

Para retenção de longo prazo de logs e análise de log, você pode baixar os arquivos .log e movê-los para um serviço terceirizado. Você pode baixar os arquivos usando o [portal do Azure](#), a [CLI do Azure](#). Como alternativa, você pode definir as configurações de diagnóstico do Azure Monitor que emitem automaticamente seus logs (no formato JSON) para locais de longo prazo. Saiba mais sobre essa opção na seção abaixo.

Você pode parar de gerar arquivos .log definindo o parâmetro `logging_collector` como OFF. É recomendável desligar a geração de arquivos .log se você estiver usando as configurações de diagnóstico do Azure Monitor. Essa configuração reduzirá o impacto no desempenho do registro em log adicional.

Logs de recursos

O Banco de Dados do Azure para PostgreSQL é integrado às configurações de diagnóstico do Azure Monitor. As configurações de diagnóstico permitem que você envie os logs do Postgres no formato JSON para os Logs do Azure Monitor para análise e alertas, Hubs de Eventos para streaming e Armazenamento do Microsoft Azure para arquivamento.

IMPORTANT

O recurso de diagnóstico para logs do servidor está disponível apenas nas [camadas de preços](#) de Uso Geral e Otimizado para Memória.

Definir as configurações de diagnóstico

Você pode habilitar as configurações de diagnóstico para o servidor Postgres usando o portal do Azure, a CLI, a API REST e o PowerShell. A categoria de log a ser selecionada é **PostgreSQLLogs**. (Há outros logs que você pode configurar se estiver usando o [Repositório de Consultas](#).)

Para habilitar os logs de recursos usando o portal do Azure:

1. No portal, vá até *Configurações de diagnóstico* no menu de navegação do servidor do Postgres.
2. Selecionar *Adicionar Configurações de Diagnóstico*.
3. Nomeie essa configuração.
4. Selecione o ponto de extremidade preferido (conta de armazenamento, hub de eventos, Log Analytics).

5. Selecione o tipo de log PostgreSQLLogs.

6. Salve sua configuração.

Para habilitar os logs de recursos usando o PowerShell, a CLI ou a API REST, confira o artigo [configurações de diagnóstico](#).

Acessar logs de recursos

A maneira como você acessa os logs depende do ponto de extremidade escolhido. Para o Armazenamento do Microsoft Azure, veja o artigo sobre [conta de armazenamento de logs](#). Para os hubs de eventos, consulte o artigo sobre [fluxos de logs do Azure](#).

Para logs de Azure Monitor, os logs são enviados para o espaço de trabalho selecionado. Os logs do Postgres usam o modo de coleta **AzureDiagnostics** para que possam ser consultados a partir da tabela **AzureDiagnostics**. Os campos na tabela são descritos abaixo. Saiba mais sobre como consultar e alertar na visão geral [Consulta de logs do Azure Monitor](#).

Para começar, execute as consultas a seguir. Você pode configurar alertas com base em consultas.

Pesquisar todos os logs do Postgres para um servidor específico no último dia

```
AzureDiagnostics  
| where LogicalServerName_s == "myservername"  
| where Category == "PostgreSQLLogs"  
| where TimeGenerated > ago(1d)
```

Pesquisar todas as tentativas de conexão não localhost

```
AzureDiagnostics  
| where Message contains "connection received" and Message !contains "host=127.0.0.1"  
| where Category == "PostgreSQLLogs" and TimeGenerated > ago(6h)
```

A consulta acima mostrará os resultados nas últimas 6 horas de qualquer log de servidor Postgres nesse workspace.

Formato de log

A tabela a seguir descreve os campos para o tipo **PostgreSQLLogs**. Dependendo do ponto de extremidade de saída escolhido, os campos incluídos e a ordem em que aparecem podem variar.

CAMPO	DESCRIÇÃO
TenantId	Sua ID de locatário
SourceSystem	Azure
TimeGenerated [UTC]	Carimbo de data/hora quando o log foi gravado, em UTC
Tipo	Tipo do log. Sempre AzureDiagnostics
SubscriptionId	GUID para a assinatura a que o servidor pertence
ResourceGroup	Nome do grupo de recursos ao qual o servidor pertence
ResourceProvider	Nome do provedor de recursos. Sempre MICROSOFT.DBFORPOSTGRESQL

CAMPO	DESCRIÇÃO
ResourceType	Servers
ResourceId	URI de recurso
Recurso	Nome do servidor
Category	PostgreSQLLogs
OperationName	LogEvent
errorLevel	Nível de log, exemplo: LOG, ERROR, NOTICE
Mensagem	Mensagem de log primária
Domínio	Versão do servidor, o exemplo: postgres-10
Detalhe	Mensagem de log secundária (se aplicável)
ColumnName	Nome da coluna (se aplicável)
SchemaName	Nome do esquema (se aplicável)
DatatypeName	Nome do tipo de dados (se aplicável)
LogicalServerName	Nome do servidor
_ResourceId	URI de recurso
Prefixo	Prefixo da linha de log

Próximas etapas

- Saiba mais sobre como acessar logs no [portal do Azure](#) ou na [CLI do Azure](#).
- Saiba mais sobre o [preço do Azure Monitor](#).
- Saiba mais sobre [logs de auditoria](#)

Log de auditoria no Banco de Dados do Azure para PostgreSQL – Servidor Único

21/05/2021 • 5 minutes to read

Log de auditoria de atividades do Banco de dados no banco de dados do Azure para PostgreSQL – um servidor único está disponível por meio da extensão de auditoria do PostgreSQL: [pgAudit](#). O pgAudit fornece o log de auditoria detalhado de sessões e/ou objetos.

NOTE

O pgAudit está em visualização no Banco de dados do Azure para PostgreSQL. A extensão pode ser habilitada somente em servidores de Uso Geral e com Otimização de Memória.

Se você quiser logs no nível de recursos do Azure para operações como o dimensionamento de computação e armazenamento, consulte o [Log de atividades do Azure](#).

Considerações sobre o uso

Por padrão, as instruções de log pgAudit são emitidas junto com suas instruções de log regulares usando o recurso de log padrão do Postgres. No banco de dados do Azure para PostgreSQL, esses arquivos.log podem ser baixados por meio do portal do Azure ou da CLI. O armazenamento máximo para a coleção de arquivos é de 1 GB e cada arquivo está disponível por um máximo de sete dias (o padrão é três dias). Esse serviço é uma opção de armazenamento de curto prazo.

Como alternativa, você pode configurar todos os logs a serem enviados para o armazenamento de log do Azure Monitor para análise posterior no Log Analytics. Se você habilitar Azure Monitor log de recursos, seus logs serão enviados automaticamente (no formato JSON) para o armazenamento do Azure, hubs de eventos e/ou logs de Azure Monitor, dependendo de sua escolha.

A habilitação de pgAudit gera um grande volume de log em um servidor, que tem um impacto no desempenho e no armazenamento de log. Recomendamos que você use os logs do Azure Monitor, que oferecem opções de armazenamento de longo prazo, bem como recursos de análise e alerta. Recomendamos que você desative o registro em log padrão para reduzir o impacto no desempenho de log adicional:

1. Defina o parâmetro `logging_collector` como Desabilitado.
2. Reinicie o servidor para aplicar suas alterações.

Para saber como configurar o log no Armazenamento do Azure, nos Hubs de Eventos ou nos logs de Azure Monitor, visite a seção de logs de recursos do [artigo de logs do servidor](#).

Como instalar o pgAudit

Para instalar o pgAudit, você precisa incluí-lo no shared preload libraries do servidor. Uma alteração no parâmetro `shared_preload_libraries` do Postgres exige uma reinicialização do servidor para que entre em vigor. Você pode alterar os parâmetros usando o [portal do Azure](#), o [CLI do Azure](#) ou a [API REST](#).

Usando o [portal do Azure](#):

1. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL.
2. Na barra lateral, selecione **Parâmetros de Servidor**.

3. Pesquise o parâmetro `shared_preload_libraries`.
4. Selecione `pgaudit`.
5. Reinicie o servidor para aplicar a alteração.
6. Conecte-se ao servidor usando um cliente (como `psql`) e habilite a extensão pgAudit.

```
CREATE EXTENSION pgaudit;
```

TIP

Se você vir um erro, confirme que você reiniciou o servidor após salvar `shared_preload_libraries`.

Configurações do pgAudit

O pgAudit permite que você configure o log de auditoria de sessão ou objeto. O [log de auditoria de sessão](#) emite logs detalhados de instruções executadas. O [log de auditoria de objeto](#) tem o escopo de auditoria para relações específicas. Você pode optar por configurar um ou ambos os tipos de registro em log.

NOTE

As configurações de pgAudit são especificadas globalmente e não podem ser especificadas em um nível de banco de dados ou função.

Quando tiver [instalado o pgAudit](#), você poderá configurar seus parâmetros para iniciar o registro em log. A [documentação do pgAudit](#) fornece a definição de cada parâmetro. Teste os parâmetros primeiro e confirme que você está obtendo o comportamento esperado.

NOTE

Definir `pgaudit.log_client` como ATIVADO redirecionará os logs para um processo de cliente, como `psql`, em vez de serem gravados no arquivo. Essa configuração deve ser deixada desabilitada.

`pgaudit.log_level` é habilitado somente quando `pgaudit.log_client` está ativado.

NOTE

No Banco de Dados do Azure para PostgreSQL, `pgaudit.log` não pode ser definido usando um atalho de sinal (menos `-`), conforme descrito na documentação do pgAudit. Todas as classes de instrução necessárias (LEITURA, GRAVAÇÃO etc.) devem ser especificadas individualmente.

Formato do log de auditoria

Cada entrada de auditoria é indicado `AUDIT:` próximo ao início da linha de log. O formato do restante da entrada é detalhado na [documentação do pgAudit](#).

Se você precisar de outros campos para atender aos seus requisitos de auditoria, use o parâmetro de Postgres `log_line_prefix`. `log_line_prefix` é uma cadeia de caracteres que é a saída no início de cada linha de log do Postgres. Por exemplo, a configuração `log_line_prefix` a seguir fornece timestamp, username, database name e process ID:

```
t=%m u=%u db=%d pid=[%p]:
```

Para saber mais sobre o `log_line_prefix`, visite a [documentação do PostgreSQL](#).

Introdução

Para começar rapidamente, defina `pgaudit.log` como `WRITE` e abra seus logs para examinar a saída.

Exibir logs de auditoria

Se estiver usando arquivos .log, os logs de auditoria serão incluídos no mesmo arquivo que os logs de erros do PostgreSQL. Você pode baixar arquivos de log do [Portal](#) ou da [CLI](#) do Azure.

Se estiver usando o log de recursos do Azure, a maneira como você acessa os logs dependerá do ponto de extremidade escolhido. Para o armazenamento do Azure, consulte o artigo sobre [conta de armazenamento de logs](#). Para os hubs de eventos, consulte o artigo sobre [fluxos de logs do Azure](#).

Para logs de Azure Monitor, os logs são enviados para o espaço de trabalho selecionado. Os logs do Postgres usam o modo de coleta [AzureDiagnostics](#) para que possam ser consultados a partir da tabela `AzureDiagnostics`. Os campos na tabela são descritos abaixo. Saiba mais sobre como consultar e alertar na visão geral [Consulta de logs do Azure Monitor](#).

Você pode usar essa consulta para começar. Você pode configurar alertas com base em consultas.

Pesquisar todos os logs do Postgres para um servidor específico no último dia

```
AzureDiagnostics
| where LogicalServerName_s == "myservername"
| where TimeGenerated > ago(1d)
| where Message contains "AUDIT:"
```

Próximas etapas

- [Saiba mais sobre o registrar em log do Banco de Dados do Azure para PostgreSQL](#).
- Saiba como configurar os parâmetros usando o [portal do Azure](#), o [CLI do Azure](#) ou a [API REST](#).

Monitorar o desempenho com o Repositório de Consultas

21/05/2021 • 11 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – Servidor Único versão 9.6 e posteriores

O recurso de Repositório de Consultas no Banco de Dados do Azure para PostgreSQL fornece uma maneira de acompanhar o desempenho de consultas ao longo do tempo. O Repositório de Consultas simplifica a solução de problemas ajudando você a rapidamente localizar as consultas de execução mais longa e que consomem mais recursos. O Repositório de Consultas captura automaticamente um histórico das estatísticas de runtime e consultas e o retém para sua análise. Ele separa os dados por janelas de tempo para que você possa ver padrões de uso do banco de dados. Os dados de todos os usuários, bancos de dados e consultas são armazenados em um banco de dados chamado `azure_sys` na instância do Banco de Dados do Azure para PostgreSQL.

IMPORTANT

Não modifique o banco de dados `azure_sys` ou seus esquemas. Fazer isso impedirá que o Repositório de Consultas e os recursos de desempenho relacionados funcionem corretamente.

Habilitando o Repositório de Consultas

O Repositório de Consultas é um recurso que requer aceitação, portanto, ele não está ativo em um servidor por padrão. O armazenamento é habilitado ou desabilitado globalmente para todos os bancos de dados em um determinado servidor e não pode ser ativado ou desativado por banco de dados.

Habilitar o Repositório de Consultas usando o portal do Azure

- Entre no portal do Azure e selecione seu servidor do Banco de Dados do Azure para PostgreSQL.
- Selecione **Parâmetros de Servidor** na seção **Configurações** do menu.
- Pesquise o parâmetro `pg_qs.query_capture_mode`.
- Defina o valor como `TOP` e **Salve**.

Para habilitar as estatísticas de espera no seu Repositório de Consultas:

- Pesquise o parâmetro `pgms_wait_sampling.query_capture_mode`.
- Defina o valor como `ALL` e **Salve**.

Como alternativa, você pode definir esse parâmetro usando a CLI do Azure.

```
az postgres server configuration set --name pg_qs.query_capture_mode --resource-group myresourcegroup --server mydemouser --value TOP
az postgres server configuration set --name pgms_wait_sampling.query_capture_mode --resource-group myresourcegroup --server mydemouser --value ALL
```

Permita que o primeiro lote de dados persista no banco de dados `azure_sys` por até 20 minutos.

Informações no Repositório de Consultas

O Repositório de Consultas tem dois repositórios:

- Um repositório de estatísticas de runtime para manter as informações de estatísticas de execução de

consulta.

- Um repositório de estatísticas de espera para manter as informações de estatísticas de execução de espera.

Os cenários comuns para usar o Repositório de Consultas incluem:

- Determinação do número de vezes que uma consulta foi executada em uma determinada janela de tempo
- Comparar o tempo médio de execução de uma consulta entre janelas de tempo para ver grandes deltas
- Identificar consultas de execução mais longas nas últimas X horas
- Identificar as principais N consultas que estão aguardando recursos
- Entender a natureza de espera de uma consulta específica

Para minimizar o uso de espaço, as estatísticas de execução de runtime no repositório de estatísticas de runtime são agregadas em uma janela de tempo fixa configurável. As informações nesses repositórios são visíveis consultando as exibições do repositório de consultas.

Acessar as informações do Repositório de Consultas

Os dados do Repositório de Consultas são armazenados no banco de dados azure_sys no servidor Postgres.

A consulta a seguir retorna informações sobre consultas no Repositório de Consultas:

```
SELECT * FROM query_store.qs_view;
```

Ou essa consulta para estatísticas de espera:

```
SELECT * FROM query_store.pgms_wait_sampling_view;
```

Localizando consultas de espera

Os tipos de evento de espera combinam diferentes eventos de espera em buckets por semelhança. O Repositório de Consultas fornece o tipo de evento de espera, o nome do evento de espera específico e a consulta em questão. Ser capaz de correlacionar essas informações de espera com as estatísticas de runtime de consulta significa que você pode obter uma compreensão mais profunda do que contribui para as características de desempenho de consulta.

Aqui estão alguns exemplos de como você pode obter mais insights sobre sua carga de trabalho usando as estatísticas de espera no Repositório de Consultas:

OBSERVAÇÃO	AÇÃO
Esperas de bloqueio alto	Verifique os textos de consulta para as consultas afetadas e identifique as entidades de destino. Procure no Repositório de Consultas outras consultas que modificam a mesma entidade, que é executada com frequência e/ou têm alta duração. Depois de identificar essas consultas, considere alterar a lógica do aplicativo para melhorar a simultaneidade ou use um nível de isolamento menos restritivo.

OBSERVAÇÃO	AÇÃO
Esperas de E/S de buffer alto	Localize as consultas com um grande número de leituras físicas no Repositório de Consultas. Se elas corresponderem às consultas com esperas de E/S altas, considere a possibilidade de introduzir um índice na entidade subjacente para realizar buscas em vez de verificações. Isso minimizaria a sobrecarga de E/S das consultas. Verifique as Recomendações de desempenho para seu servidor no portal para ver se há recomendações de índice para esse servidor que otimizariam as consultas.
Esperas de memória alta	Localize as consultas que consomem mais memória no Repositório de Consultas. Essas consultas estão provavelmente atrasando o andamento das consultas afetadas. Verifique as Recomendações de desempenho para seu servidor no portal para ver se há recomendações de índice que otimizariam essas consultas.

Opções de configuração

Quando o Repositório de Consultas está habilitado, ele salva dados em janelas de agregação de 15 minutos, até 500 consultas distintas por janela.

As opções a seguir estão disponíveis para configurar os parâmetros do Repositório de Consultas.

PARÂMETRO	DESCRÍÇÃO	DEFAULT	RANGE
pg_qs.query_capture_mode	Define quais instruções são rastreadas.	nenhum	none, top, all
pg_qs.max_query_text_length	Define o comprimento máximo de consulta que pode ser salvo. Consultas mais longas serão truncadas.	6000	100 a 10 mil
pg_qs.retention_period_in_days	Define o período de retenção.	7	1 a 30
pg_qs.track_utility	Define se os comandos do utilitário são rastreados	on	on, off

As opções a seguir se aplicam especificamente às estatísticas de espera.

PARÂMETRO	DESCRÍÇÃO	DEFAULT	RANGE
pgms_wait_sampling.query_capture_mode	Define quais instruções são rastreadas para as estatísticas de espera.	nenhum	none, all
Pgms_wait_sampling.history_period	Define a frequência, em milissegundos, com a qual são realizadas amostras dos eventos de espera.	100	1 a 600000

NOTE

`pg_qs.query_capture_mode` substitui `pgms_wait_sampling.query_capture_mode`. Se `pg_qs.query_capture_mode` for `NONE`, a configuração `pgms_wait_sampling.query_capture_mode` não terá efeito.

Use o [portal do Azure](#) ou a [CLI do Azure](#) para obter ou definir um valor diferente para um parâmetro.

Exibições e funções

Exiba e gerencie o Repositório de Consultas usando as seguintes exibições e funções. Qualquer pessoa na função pública do PostgreSQL pode usar essas exibições para ver os dados no Repositório de Consultas. Essas exibições estão disponíveis somente no banco de dados `azure_sys`.

Consultas são normalizadas examinando sua estrutura após a remoção de literais e constantes. Se duas consultas forem idênticas, exceto por valores literais, elas terão o mesmo hash.

`query_store.qs_view`

Essa exibição retorna todos os dados no Repositório de Consultas. Há uma linha para cada ID de banco de dados, ID de usuário e ID de consulta distinta.

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
<code>runtime_stats_entry_id</code>	BIGINT		ID da tabela <code>runtime_stats_entries</code>
<code>user_id</code>	oid	<code>pg_authid.oid</code>	OID do usuário que executou a instrução
<code>db_id</code>	oid	<code>pg_database.oid</code>	OID do banco de dados no qual a instrução foi executada
<code>query_id</code>	BIGINT		Código hash interno, computado da árvore de análise da instrução
<code>query_sql_text</code>	Varchar(10000)		Texto de uma instrução representativa. Consultas diferentes com a mesma estrutura são agrupadas. Este texto é o da primeira das consultas no cluster.
<code>plan_id</code>	BIGINT		ID do plano correspondente a essa consulta, ainda não disponível
<code>start_time</code>	timestamp		Consultas são agregadas por buckets de tempo: o período de um bucket é de 15 minutos por padrão. Essa é a hora de início correspondente ao bucket de tempo para esta entrada.

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
end_time	timestamp		Hora de término correspondente ao bucket de tempo para esta entrada.
chamadas	BIGINT		Número de vezes que a consulta foi executada
total_time	double precision		Tempo total de execução da consulta em milissegundos
min_time	double precision		Tempo mínimo de execução da consulta em milissegundos
max_time	double precision		Tempo máximo de execução da consulta em milissegundos
mean_time	double precision		Tempo médio de execução da consulta em milissegundos
stddev_time	double precision		Desvio padrão de tempo de execução da consulta em milissegundos
rows	BIGINT		Número total de linhas recuperadas ou afetadas pela instrução
shared_blk_hit	BIGINT		Número total de ocorrências no cache do bloco compartilhado pela instrução
shared_blk_read	BIGINT		Número total de blocos compartilhados lidos pela instrução
shared_blk_dirtied	BIGINT		Número total de blocos compartilhados sujos pela instrução
shared_blk_written	BIGINT		Número total de blocos compartilhados gravados pela instrução
local_blk_hit	BIGINT		Número total de ocorrências no cache do bloco local pela instrução
local_blk_read	BIGINT		Número total de leituras de blocos locais pela instrução

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
local_blks_dirtied	BIGINT		Número total de blocos locais sujos pela instrução
local_blks_written	BIGINT		Número total de blocos locais gravados pela instrução
temp_blks_read	BIGINT		Número total de leituras de blocos temporários pela instrução
temp_blks_written	BIGINT		Número total de gravações de blocos temporários pela instrução
blk_read_time	double precision		Tempo total que a instrução passou lendo blocos em milissegundos (se track_io_timing estiver habilitado, caso contrário, zero)
blk_write_time	double precision		Tempo total que a instrução passou gravando blocos em milissegundos (se track_io_timing estiver habilitado, caso contrário, zero)

query_store.query_texts_view

Essa exibição retorna os dados de texto da consulta no Repositório de Consultas. Há uma linha para cada query_text distinto.

NOME	TIPO	DESCRIÇÃO
query_text_id	BIGINT	ID da tabela query_texts
query_sql_text	Varchar(10000)	Texto de uma instrução representativa. Consultas diferentes com a mesma estrutura são agrupadas. Este texto é o da primeira das consultas no cluster.

query_store.pgms_wait_sampling_view

Essa exibição retorna os dados de eventos de espera no Repositório de Consultas. Há uma linha para cada ID de banco de dados, ID de usuário, ID de consulta e evento distinto.

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
user_id	oid	pg_authid.oid	OID do usuário que executou a instrução
db_id	oid	pg_database.oid	OID do banco de dados no qual a instrução foi executada

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
query_id	BIGINT		Código hash interno, computado da árvore de análise da instrução
event_type	text		O tipo de evento pelo qual o back-end está esperando
event	text		O nome do evento de espera se o back-end estiver esperando no momento
chamadas	Integer		Número do mesmo evento capturado

Funções

Query_store.qs_reset() retorna void

`qs_reset` descarta todas as estatísticas coletadas até o momento pelo Repositório de Consultas. Essa função só pode ser executada pela função de administrador de servidor.

Query_store.staging_data_reset() retorna void

`staging_data_reset` descarta todas as estatísticas coletadas na memória pelo Repositório de Consultas (isto é, os dados na memória que ainda não foram liberados para o banco de dados). Essa função só pode ser executada pela função de administrador de servidor.

Azure Monitor

O Banco de Dados do Azure para PostgreSQL é integrado às [configurações de diagnóstico do Azure Monitor](#). As configurações de diagnóstico permitem que você envie os logs do Postgres no formato JSON para os [Logs do Azure Monitor](#) para análise e alertas, Hubs de Eventos para streaming e Armazenamento do Microsoft Azure para arquivamento.

IMPORTANT

O recurso de diagnóstico está disponível apenas nas camadas de preços de Uso geral e de Otimizado para memória.

Definir as configurações de diagnóstico

Você pode habilitar as configurações de diagnóstico para o servidor Postgres usando o portal do Azure, a CLI, a API REST e o PowerShell. As categorias de log a configurar são `QueryStoreRuntimeStatistics` e `QueryStoreWaitStatistics`.

Para habilitar os logs de recursos usando o portal do Azure:

1. No portal, acesse Configurações de Diagnóstico no menu de navegação do servidor Postgres.
2. Selecione Adicionar configuração de diagnóstico.
3. Nomeie essa configuração.
4. Selecione o ponto de extremidade preferido (conta de armazenamento, hub de eventos, Log Analytics).
5. Selecione os tipos de log `QueryStoreRuntimeStatistics` e `QueryStoreWaitStatistics`.
6. Salve sua configuração.

Para habilitar essa configuração usando o PowerShell, a CLI ou a API REST, veja o [artigo de configurações de](#)

diagnóstico.

Formato de log JSON

As tabelas a seguir descrevem os campos para os dois tipos de log. Dependendo do ponto de extremidade de saída escolhido, os campos incluídos e a ordem em que aparecem podem variar.

QueryStoreRuntimeStatistics

CAMPO	DESCRIÇÃO
TimeGenerated [UTC]	Carimbo de data/hora quando o log foi gravado, em UTC
ResourceId	URI de recurso do Azure do servidor Postgres
Categoria	QueryStoreRuntimeStatistics
OperationName	QueryStoreRuntimeStatisticsEvent
LogicalServerName_s	Nome do servidor Postgres
runtime_stats_entry_id_s	ID da tabela runtime_stats_entries
user_id_s	OID do usuário que executou a instrução
db_id_s	OID do banco de dados no qual a instrução foi executada
query_id_s	Código hash interno, computado da árvore de análise da instrução
end_time_s	Hora de término correspondente ao bucket de tempo para esta entrada
calls_s	Número de vezes que a consulta foi executada
total_time_s	Tempo total de execução da consulta em milissegundos
min_time_s	Tempo mínimo de execução da consulta em milissegundos
max_time_s	Tempo máximo de execução da consulta em milissegundos
mean_time_s	Tempo médio de execução da consulta em milissegundos
ResourceGroup	O grupo de recursos
SubscriptionId	Sua ID de assinatura
ResourceProvider	Microsoft.DBForPostgreSQL
Recurso	Nome do servidor Postgres
ResourceType	Servers

QueryStoreWaitStatistics

CAMPO	DESCRIÇÃO
TimeGenerated [UTC]	Carimbo de data/hora quando o log foi gravado, em UTC
ResourceId	URI de recurso do Azure do servidor Postgres
Categoria	QueryStoreWaitStatistics
OperationName	QueryStoreWaitEvent
user_id_s	OID do usuário que executou a instrução
db_id_s	OID do banco de dados no qual a instrução foi executada
query_id_s	Código hash interno da consulta
calls_s	Número do mesmo evento capturado
event_type_s	O tipo de evento pelo qual o back-end está esperando
event_s	O nome do evento de espera se o back-end estiver esperando no momento
start_time_t	Hora de início do evento
end_time_s	Hora de término do evento
LogicalServerName_s	Nome do servidor Postgres
ResourceGroup	O grupo de recursos
SubscriptionId	Sua ID de assinatura
ResourceProvider	Microsoft.DBForPostgreSQL
Recurso	Nome do servidor Postgres
ResourceType	Servers

Limitações e problemas conhecidos

- Se um servidor PostgreSQL tem o parâmetro default_transaction_read_only ativo, o Repositório de Consultas não é capaz de capturar dados.
- A funcionalidade do Repositório de Consultas poderá ser interrompida se ele encontrar consultas Unicode longas (> = 6.000 bytes).
- As [réplicas de leitura](#) replicam os dados do Repositório de Consultas do servidor primário. Isso significa que o Repositório de Consultas de uma réplica de leitura não fornece estatísticas sobre as consultas executadas na réplica de leitura.

Próximas etapas

- Saiba mais sobre [cenários em que o Repositório de Consultas pode ser especialmente útil](#).

- Saiba mais sobre as [melhores práticas para usar o Repositório de Consultas](#).

Cenários de uso de Store de consulta

09/08/2021 • 3 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – versões 9.6, 10, 11 do servidor único

Você pode usar o Query Store em uma ampla variedade de cenários, nos quais o acompanhamento e a manutenção do desempenho previsível da carga de trabalho são essenciais. Considere os seguintes exemplos:

- Identificar e ajustar consultas caras superior
- Testes de A/B
- Mantendo o desempenho estável durante as atualizações
- Identificando e melhorando as cargas de trabalho ad hoc

Identificar e ajustar consultas caras

Identificar as consultas de execução mais longas

Use a exibição [Query Performance Insight](#) no portal do Azure para identificar rapidamente as consultas mais longas. Essas consultas normalmente tendem a consumir uma quantidade significativa de recursos. A otimização de suas perguntas mais longas pode melhorar o desempenho, liberando recursos para uso por outras consultas em execução no seu sistema.

Consultas de destino com deltas de desempenho

O Query Store divide os dados de desempenho em janelas de tempo, para que você possa acompanhar o desempenho de uma consulta ao longo do tempo. Isso ajuda a identificar exatamente quais consultas estão contribuindo para um aumento no tempo total gasto. Como resultado, você pode fazer uma solução de problemas direcionada de sua carga de trabalho.

Ajuste de consultas caras

Quando você identifica uma consulta com desempenho abaixo do ideal, a ação executada depende da natureza do problema:

- Use [Recomendações de desempenho](#) para determinar se há algum índice sugerido. Se sim, crie o índice e use o Query Store para avaliar o desempenho da consulta depois de criar o índice.
- Certifique-se de que as estatísticas estejam atualizadas para as tabelas subjacentes usadas pela consulta.
- Pense em reescrever as consultas caras. Por exemplo, aproveite a parametrização de consulta e reduza o uso de SQL dinâmico. Implemente a lógica ideal ao ler dados, como aplicar a filtragem de dados no lado do banco de dados, não no lado do aplicativo.

Testes de A/B

Use o Query Store para comparar o desempenho da carga de trabalho antes e depois de uma alteração de aplicativo que você planeja introduzir. Exemplos de cenários para usar o Query Store para avaliar o impacto do ambiente ou a alteração do aplicativo no desempenho da carga de trabalho:

- Implantando uma nova versão de um aplicativo.
- Adicionando recursos adicionais para o servidor.
- Criando índices ausentes em tabelas referenciadas por consultas caras.

Em qualquer um desses cenários, aplique o seguinte fluxo de trabalho:

1. Execute sua carga de trabalho com o Query Store antes da mudança planejada para gerar uma linha de base

- de desempenho.
2. Aplique mudanças de aplicação no momento controlado no tempo.
 3. Continue executando a carga de trabalho por tempo suficiente para gerar uma imagem de desempenho do sistema após a alteração.
 4. Compare os resultados de antes e depois da alteração.
 5. Decida se deseja manter a alteração ou reversão.

Identifique e melhore as cargas de trabalho ad hoc

Algumas cargas de trabalho não possuem consultas dominantes que você pode ajustar para melhorar o desempenho geral do aplicativo. Essas cargas de trabalho normalmente são caracterizadas com um número relativamente grande de consultas exclusivas, cada uma consumindo uma parte dos recursos do sistema. Cada consulta exclusiva é executada com pouca frequência, portanto, individualmente, seu consumo em runtime não é crítico. Por outro lado, como o aplicativo está gerando novas consultas o tempo todo, uma parte significativa dos recursos do sistema é gasta na compilação de consultas, o que não é o ideal. Normalmente, essa situação acontece se o aplicativo gerar consultas (em vez de usar procedimentos armazenados ou consultas parametrizadas) ou se depender de estruturas de mapeamento objeto-relacional que geram consultas por padrão.

Se você estiver no controle do código do aplicativo, considere reescrever a camada de acesso a dados para usar procedimentos armazenados ou consultas parametrizadas. No entanto, essa situação também pode ser melhorada sem alterações de aplicativo, forçando a parametrização de consulta para todo o banco de dados (todas as consultas) ou para os modelos de consulta individuais com o mesmo hash de consulta.

Próximas etapas

- Saiba mais sobre o [práticas recomendadas para usar a consulta Store](#)

Práticas recomendadas para Repositório de Consultas

26/05/2021 • 2 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – versões 9.6, 10, 11 do servidor individual

Este artigo descreve as práticas recomendadas para o uso do Repositório de Consultas no Banco de Dados do Azure para PostgreSQL.

Definir o modo de captura de consulta ideal

Deixe que o Repositório de Consultas capture os dados que importam para você.

PG_QS.QUERY_CAPTURE_MODE	CENÁRIO
<i>Todos</i>	Analise sua carga de trabalho cuidadosamente em termos de todas as consultas e das respectivas frequências de execução e outras estatísticas. Identifique novas consultas na carga de trabalho. Detecte se consultas ad hoc são usadas para identificar oportunidades de parametrização automática ou pelo usuário. <i>All</i> acompanha um custo de consumo de recursos maior.
<i>Top</i>	Concentre sua atenção nas principais consultas – aquelas emitidas pelos clientes.
<i>Nenhuma</i>	Você já capturou um conjunto de consultas e a janela de tempo que você deseja investigar, e você deseja eliminar as distrações que outras consultas podem causar. <i>None</i> é adequado para teste e avaliação de desempenho de ambientes. <i>None</i> deve ser usado com cuidado, pois você pode perder a oportunidade de acompanhar e otimizar consultas novas importantes. Você não pode recuperar dados nessas janelas de tempo do passado.

O Repositório de Consultas também inclui um repositório de estatísticas de espera. Há uma consulta de modo de captura adicional que controla as estatísticas de espera: `pgms_wait_sampling.query_capture_mode` pode ser definido como *none* ou *all*.

NOTE

`pg_qs.query_capture_mode` substitui `pgms_wait_sampling.query_capture_mode`. Se `pg_qs.query_capture_mode` for *none*, a configuração `pgms_wait_sampling.query_capture_mode` não terá efeito.

Manter os dados de que precisa

O parâmetro `pg_qs.retention_period_in_days` especifica, em dias, o período de retenção de dados para o Repositório de Consultas. Dados de consulta e estatísticas mais antigos são excluídos. Por padrão, o Repositório de Consultas é configurado para reter os dados por 7 dias. Evite manter dados históricos que você não planeja usar. Aumente o valor se você precisar manter dados por mais tempo.

Definir a frequência de amostragem de estatísticas de espera

O parâmetro `pgms_wait_sampling.history_period` especifica a frequência (em milissegundos) em que eventos de espera são usados como amostra. Quanto menor for o período, mais frequente será a amostragem. Mais informações são recuperadas, mas isso acompanha o custo de um maior consumo de recursos. Aumente esse período se o servidor está sob carregamento ou se você não precisa de granularidade.

Obter insights rápidos sobre o Repositório de Consultas

Você pode usar a [Análise de Desempenho de Consultas](#) no portal do Azure para obter insights rápidos sobre os dados no Repositório de Consultas. As visualizações revelam as consultas em execução a mais tempo e os eventos de espera de maior duração ao longo do tempo.

Próximas etapas

- Saiba como obter ou definir parâmetros usando o [portal do Azure](#) ou a [CLI do Azure](#).

Análise de Desempenho de Consultas

09/08/2021 • 2 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – versões 9.6, 10, 11 do servidor único

A Análise de Desempenho de Consultas ajuda você a identificar rapidamente quais são suas consultas de execução mais longa, como elas mudam ao longo do tempo e quais esperas as estão afetando.

Permissões

Permissões do Proprietário ou Colaborador necessárias para exibir o texto das consultas na Análise de Desempenho de Consultas **Leitor** podem exibir gráficos e tabelas, mas não o texto da consulta.

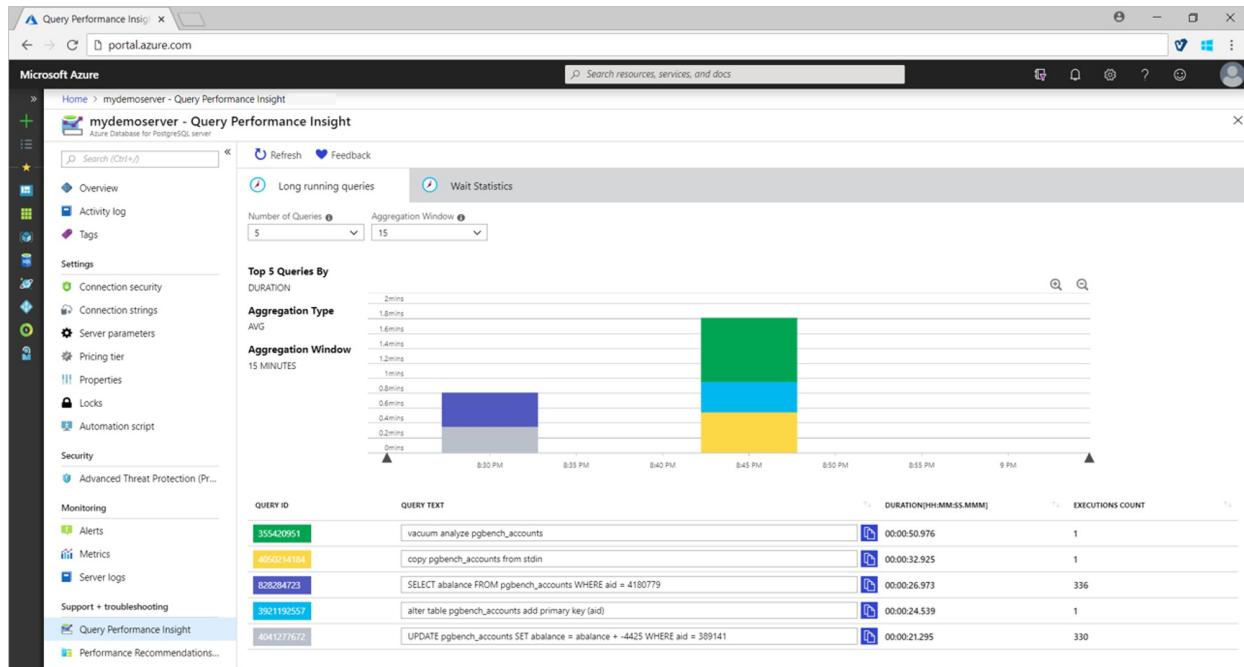
Pré-requisitos

Para a Análise de Desempenho de Consultas funcionar, os dados precisam existir no [Repositório de Consultas](#).

Exibição de análises de desempenho

A visualização da [Análise de Desempenho de Consultas](#) no portal do Azure será superficial visualizações em informações do Repositório de Consultas.

Na página do portal do servidor do Banco de Dados do Azure para PostgreSQL, selecione **Análise de Desempenho de Consultas** na seção **Desempenho Inteligente** da barra de menus.

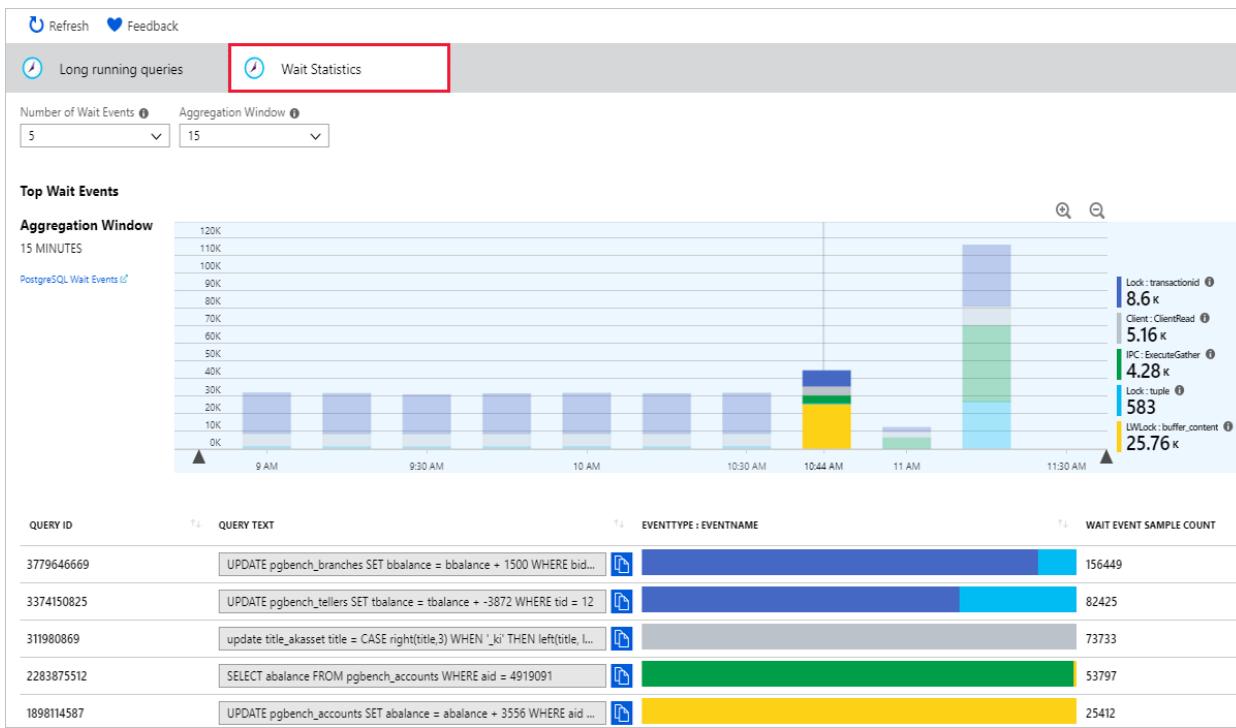


A guia **Consultas de execução prolongada** mostra as cinco principais consultas por duração média por execução, agregadas em intervalos de 15 minutos. Você pode exibir mais consultas, selecionando a partir do **Número de consultas** lista suspensa. As cores do gráfico pode ser alteradas para uma ID de consulta específica ao fazer isso.

Você pode clicar e arrastar no gráfico para restringi-lo a uma janela de tempo específico. Como alternativa, use os ícones de ampliar e afastar para exibir um período maior ou menor, respectivamente.

A tabela abaixo do gráfico contém mais detalhes sobre as consultas de execução longa na janela de tempo.

Selecione a guia das **Estatísticas de Espera** guia para exibir as visualizações correspondentes em espera no servidor.



Considerações

- A Análise de Desempenho de Consultas não está disponível para [réplicas de leitura](#).

Próximas etapas

- Saiba mais sobre [monitoramento e ajuste](#) no Banco de Dados do Azure para PostgreSQL.

Recomendações de desempenho do Banco de Dados do Azure para PostgreSQL – Servidor único

09/08/2021 • 2 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – versões 9.6, 10, 11 do servidor único

O recurso Recomendações de desempenho analisa os bancos de dados para criar sugestões personalizadas para desempenho aprimorado. Para produzir as recomendações, a análise examina várias características do banco de dados, incluindo o esquema. Habilite o [Repositório de Consultas](#) no servidor para utilizar totalmente o recurso Recomendações de Desempenho. Depois de implementar qualquer recomendação de desempenho, você deve testar o desempenho para avaliar o impacto dessas alterações.

Permissões

Permissões de Proprietário ou Colaborador necessárias para executar a análise usando o recurso de recomendações de desempenho.

Recomendações do desempenho

O recurso das [Recomendações de Desempenho](#) recurso analisa as cargas de trabalho entre seu servidor para identificar os índices com o potencial de melhorar o desempenho.

Abra as **Recomendações de desempenho** da seção **Desempenho inteligente** na barra de menus da página do portal do Azure do servidor PostgreSQL.

The screenshot shows the Microsoft Azure portal interface. The URL in the address bar is portal.azure.com/. The main content area is titled "mydemoserver - Performance Recommendations" and "Azure Database for PostgreSQL server". At the top right, there are buttons for "Analyze" (highlighted with a red box), "Refresh", and "Feedback". Below the title, there's a search bar and a "Recommendations" table with columns: ACTION, DATE, DATABASE, INDEX SIZE, RECOMMENDATION DESCRIPTION, and IMPACT. A message states: "There are currently no performance recommendations to display. Once you perform analysis on a database, its results will be displayed here." On the left side, there's a navigation sidebar with sections like Tags, Diagnose and solve problems, Settings, Security, Intelligent Performance, and Monitoring. Under Intelligent Performance, the "Performance Recommendations" item is also highlighted with a red box.

Selecione **Analisar** e escolha um banco de dados que iniciará a análise. Dependendo da carga de trabalho, a análise poderá levar vários minutos para ser concluída. Quando a análise for concluída, haverá uma notificação no portal. A análise executa um exame profundo do banco de dados. Recomendamos que você execute a análise fora dos períodos de pico.

A janela Recomendações de desempenho exibirá uma lista de recomendações, caso sejam encontradas.

Recommendations						IMPACT
ACTION	DATE	DATABASE	INDEX SIZE	RECOMMENDATION DESCRIPTION	IMPACT	
CREATE INDEX	02/04/2019	testdb	6.23 MB	Create index on 'public.sometable(id)'.	High	
DROP INDEX	02/04/2019	testdb	5.00 MB	Drop index 'anothertable_idx2' on table 'public.anothertable'. Reason: Index is a duplicate of 'anothertable_idx1'.	Medium	

As recomendações não são aplicadas automaticamente. Para aplicar a recomendação, copie o texto da consulta e execute-o no cliente de sua escolha. Lembre-se de testar e monitorar para avaliar a recomendação.

Tipos de recomendação

No momento, há suporte para dois tipos de recomendações: *Criar índice* e *Remover índice*.

Criar recomendações de índice

As recomendações *Criar Índice* sugerem novos índices para acelerar as consultas demoradas ou executadas com mais frequência na carga de trabalho. Esse tipo de recomendação requer a habilitação do [Repositório de Consultas](#). O Repositório de Consultas coleta informações de consulta e fornece estatísticas detalhadas de runtime e frequência de consulta usadas pela análise para fazer a recomendação.

Recomendações para Remover Índice

Além de detectar índices ausentes, o Banco de Dados do Azure para PostgreSQL analisa o desempenho dos índices existentes. Se um índice for raramente usado ou for redundante, o analisador recomenda sua remoção.

Considerações

- As Recomendações de desempenho não estão disponíveis para [réplicas de leitura](#).

Próximas etapas

- Saiba mais sobre [monitoramento e ajuste](#) no Banco de Dados do Azure para PostgreSQL.

Assistente do Azure para PostgreSQL

21/05/2021 • 2 minutes to read

Saiba mais sobre como o Assistente do Azure é aplicado ao Banco de Dados do Azure para PostgreSQL e obtenha respostas para perguntas comuns.

O que é o Assistente do Azure para PostgreSQL?

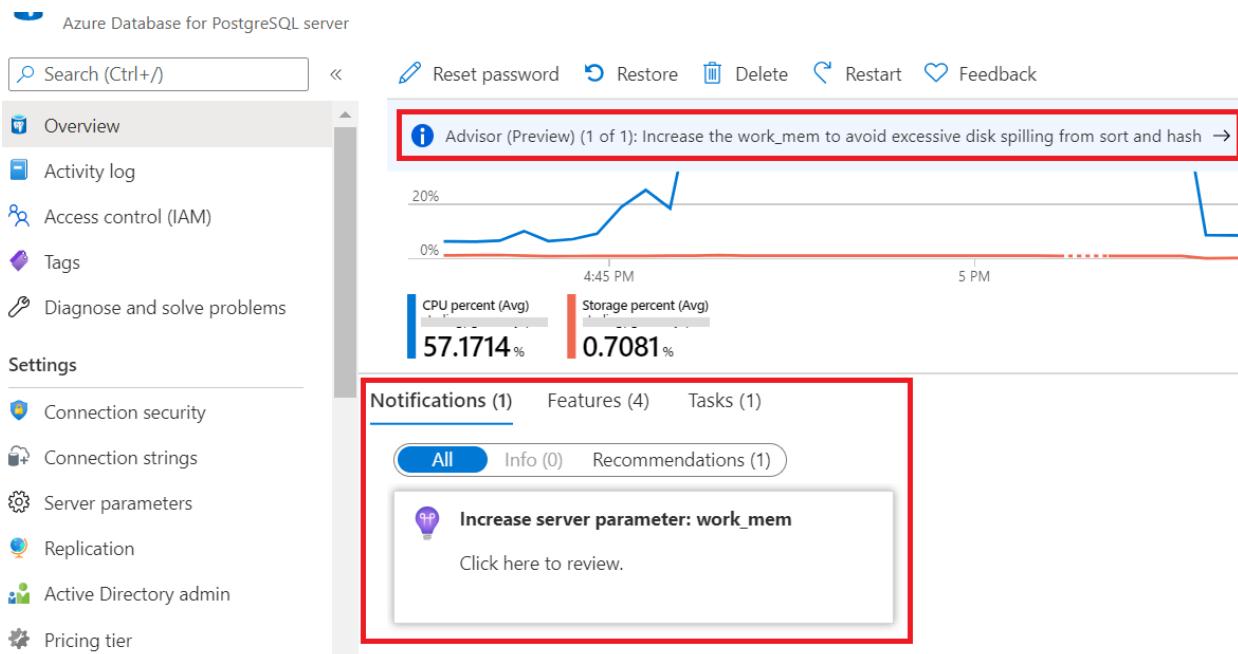
O sistema do Assistente do Azure usa telemetria para emitir recomendações de desempenho e confiabilidade para o banco de dados PostgreSQL. As recomendações do Assistente estão divididas entre as nossas ofertas de banco de dados PostgreSQL:

- Banco de Dados do Azure para PostgreSQL – Servidor único
- Servidor Flexível do Banco de Dados do Azure para PostgreSQL
- Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

Algumas recomendações são comuns a várias ofertas de produtos, enquanto outras são baseadas em otimizações específicas de produtos.

Onde posso exibir minhas recomendações?

As recomendações estão disponíveis na barra lateral de navegação **Visão geral** no portal do Azure. Uma visualização será exibida como uma faixa de notificação e os detalhes poderão ser exibidos na seção **Notificações** localizada logo abaixo dos gráficos de uso de recursos.



Tipos de recomendação

O Banco de Dados do Azure para PostgreSQL prioriza os seguintes tipos de recomendação:

- **Desempenho:** para melhorar a velocidade do servidor PostgreSQL. Isso inclui uso de CPU, pressão de memória, pool de conexões, utilização de disco e parâmetros de servidor específicos do produto. Para saber mais, veja [Advisor Performance recommendations](#) (Recomendações de desempenho do Advisor).
- **Confiabilidade:** para garantir e melhorar a continuidade dos bancos de dados críticos para os negócios.

Isso inclui limites de armazenamento, limites de conexão e recomendações de distribuição de dados de hiperescala. Para saber mais, consulte [Advisor Reliability recommendations](#) (Recomendações de confiabilidade do Assistente).

- **Custo:** para otimizar e reduzir o gasto geral do Azure. Isso inclui recomendações de dimensionamento correto do servidor. Para saber mais, veja [Advisor Cost recommendations](#) (Recomendações de custo do Advisor).

Entendendo as recomendações

- **Agendamento diário:** para Bancos de Dados PostgreSQL do Azure, verificamos a telemetria do servidor e as recomendações de problemas para um agendamento diário. Caso as configurações do servidor sejam alteradas, as recomendações existentes permanecerão visíveis até a telemetria ser examinada novamente, no dia seguinte.
- **Histórico de desempenho:** algumas recomendações são baseadas no histórico de desempenho. Essas recomendações só serão exibidas depois que um servidor estiver operando com a mesma configuração por 7 dias. Isso permite detectar padrões de uso intenso (por exemplo, alta atividade de CPU ou alto volume de conexão) em um período de tempo sustentado. Se um novo servidor for provisionado ou alterado para uma nova configuração de vCore, essas recomendações são pausadas temporariamente. Isso impede que a telemetria herdada dispare recomendações em um servidor reconfigurado recentemente. No entanto, isso também significa que as recomendações baseadas no histórico de desempenho podem não ser identificadas imediatamente.

Próximas etapas

Para obter mais informações, consulte [Visão Geral do Assistente do Azure](#).

Melhores práticas para criar um aplicativo com o Banco de Dados do Azure para PostgreSQL

21/05/2021 • 6 minutes to read

Aqui estão algumas melhores práticas para ajudá-lo a criar um aplicativo pronto para a nuvem usando o Banco de Dados do Azure para PostgreSQL. Essas melhores práticas podem reduzir o tempo de desenvolvimento para seu aplicativo.

Configuração de recursos de aplicativo e banco de dados

Mantenha o aplicativo e o banco de dados na mesma região

Verifique se todas as suas dependências estão na mesma região ao implantar seu aplicativo no Azure. A difusão de instâncias entre regiões ou zonas de disponibilidade cria a latência de rede, o que pode afetar o desempenho geral do seu aplicativo.

Mantenha seu servidor PostgreSQL seguro

Configure seu servidor PostgreSQL para ser [seguro](#) e não acessível publicamente. Use uma destas opções para proteger seu servidor:

- [Regras de firewall](#)
- [Redes virtuais](#)
- [Link Privado do Azure](#)

Para segurança, você deve sempre se conectar ao servidor PostgreSQL por SSL e configurar seu servidor PostgreSQL e seu aplicativo para usar o TLS 1.2. Confira [Como configurar o SSL/TLS](#).

Ajustar os parâmetros do servidor

Para cargas de trabalho de leitura pesadas que ajustam parâmetros de servidor, `tmp_table_size` e `max_heap_table_size` podem ajudar a otimizar para melhor desempenho. Para calcular os valores necessários para essas variáveis, revise o total de valores de memória por conexão e a memória base. A soma dos parâmetros de memória por conexão, exceto `tmp_table_size`, combinada com as contas de memória base para a memória total do servidor.

Usar variáveis de ambiente para informações de conexão

Não salve suas credenciais de banco de dados no código do aplicativo. Dependendo do aplicativo de front-end, siga as diretrizes para configurar as variáveis de ambiente. No uso do serviço de aplicativo, consulte [Como definir configurações de aplicativo](#) e no serviço Kubernetes do Azure, consulte [Como usar segredos do Kubernetes](#).

Desempenho e resiliência

Aqui estão algumas ferramentas e práticas que você pode usar para ajudar a depurar problemas de desempenho com seu aplicativo.

Usar o pool de conexões

Com o pool de conexões, um conjunto fixo de conexões é estabelecido no momento da inicialização e mantido. Isso também ajuda a reduzir a fragmentação de memória no servidor que é causado pelas conexões dinâmicas novas estabelecidas no servidor de banco de dados. O pooling de conexão poderá ser configurado do lado do aplicativo se a estrutura do aplicativo ou o driver de banco de dados oferecer suporte a ele. Se ele não for

suportado, a outra opção recomendada é aproveitar um serviço de pooler de conexão de proxy como [PgBouncer](#) ou [Pgpool](#) em execução fora do aplicativo e conectar-se ao servidor de banco de dados. O PgBouncer e o Pgpool são ferramentas baseadas na comunidade que funcionam com o Banco de Dados do Azure para PostgreSQL.

Lógica de repetição para manipular erros transitórios

Seu aplicativo pode apresentar erros transitórios nos quais as conexões com o banco de dados são descartadas ou perdidas de forma intermitente. Nessas situações, o servidor está em execução após uma ou duas repetições em 5 a 10 segundos. Uma boa prática é aguardar 5 segundos antes de sua primeira tentativa. Em seguida, siga cada nova tentativa aumentando a espera gradualmente, até 60 segundos. Limite o número máximo de repetições em que o seu aplicativo considera que a operação falhou, para que você possa investigar ainda mais. Confira [Como solucionar problemas de erros de conexão](#) para saber mais.

Habilitar a replicação de leitura para reduzir failovers

Você pode usar a [Replicação de Dados](#) para cenários de failover. Quando você está usando réplicas de leitura, não ocorre nenhum failover automatizado entre os servidores de origem e de réplica. Você perceberá um atraso entre a origem e a réplica porque a replicação é assíncrona. O atraso da rede pode ser influenciado por muitos fatores, como o tamanho da carga de trabalho em execução no servidor de origem e a latência entre os data centers. Na maioria dos casos, o atraso da réplica varia de alguns segundos para alguns minutos.

Implantação de banco de dados

Configurar o pipeline de implantação de CI/CD

Ocasionalmente, você precisa implantar alterações em seu banco de dados. Nesses casos, você pode usar a CI (integração contínua) por meio de [ações do GitHub](#) para o servidor PostgreSQL atualizar o banco de dados executando um script personalizado em relação a ele.

Definir processo de implantação manual de banco de dados

Durante a implantação manual do banco de dados, siga estas etapas para minimizar o tempo de inatividade ou reduzir o risco de falha na implantação:

- Crie uma cópia de um banco de dados de produção em um novo banco de dados usando pg_dump.
- Atualize o novo banco de dados com suas novas alterações de esquema ou atualizações necessárias para seu banco de dados.
- Coloque o banco de dados de produção em um estado somente leitura. Você não deve ter operações de gravação no banco de dados de produção até que a implantação seja concluída.
- Teste seu aplicativo com o banco de dados recentemente atualizado da etapa 1.
- Implante as alterações do aplicativo e verifique se o aplicativo agora está usando o novo banco de dados que tem as atualizações mais recentes.
- Mantenha o banco de dados de produção antigo para que você possa reverter as alterações. Em seguida, você pode avaliar excluir o banco de dados de produção antigo ou exportá-lo no Armazenamento do Azure, se necessário.

[!NOTE] Se o aplicativo for como um aplicativo de comércio eletrônico e você não puder colocá-lo no estado somente leitura, implante as alterações diretamente no banco de dados de produção depois de fazer um backup. Essas alterações devem ocorrer fora do horário de pico, com baixo tráfego no aplicativo para minimizar o impacto, pois alguns usuários podem ter solicitações com falha. Verifique se o código do aplicativo também trata das solicitações com falha.

Esquema e consultas do banco de dados

Aqui estão algumas dicas para não esquecer quando você criar seu esquema de banco de dados e consultas.

Usar BIGINT ou UUID para Chaves Primárias

Ao criar aplicativos personalizados ou algumas estruturas eles podem usar `INT` em vez de `BIGINT` para chaves primárias. Ao usar `INT`, você corre o risco do valor em seu banco de dados exceder a capacidade de armazenamento do tipo de dado `INT`. Fazer essa alteração em um aplicativo de produção existente pode ser demorado custando mais tempo de desenvolvimento. Outra opção é usar o `UUID` para chaves primárias. Esse identificador usa uma cadeia de caracteres de 128 bits gerada automaticamente, por exemplo `a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11`. Saiba mais sobre os [Tipos de dados PostgreSQL](#).

Usar índices

Há muitos tipos de [índices](#) no Postgres que podem ser usados de formas diferentes. O uso de um índice ajuda o servidor a localizar e recuperar linhas específicas muito mais rápido do que poderia sem um índice. Mas os índices também adicionam sobrecarga ao servidor de banco de dados, portanto, evite ter muitos índices.

Usar vácuo automático

Você pode otimizar seu servidor com o vácuo automático em um servidor de Banco de Dados PostgreSQL do Azure. O PostgreSQL permite maior simultaneidade de banco de dados, mas cada atualização resulta em inserção e exclusão. Para excluir, os registros marcados temporariamente serão limpos mais tarde. Para executar essas tarefas, o PostgreSQL executa um trabalho de vácuo. Se você não executar o vácuo periodicamente, as tuplas inativas acumuladas poderão levar a:

- Sobrelocação de dados, como bancos de dados e tabelas maiores.
- Maiores índices de qualidade inferior.
- Aumento de E/S.

Saiba mais sobre [Como otimizar com o vácuo automático](#).

Usar pg_stats_statements

`Pg_stat_statements` é uma extensão do PostgreSQL habilitada por padrão no Banco de Dados do Azure para PostgreSQL. A extensão fornece um modo para rastrear as estatísticas de execução de todas as instruções SQL executadas por um servidor. Confira [Como usar o pg_statement](#).

Usar o Repositório de Consultas

O recurso [Repositório de Consultas](#) no Banco de Dados do Azure para PostgreSQL fornece um método mais eficaz para rastrear as estatísticas da consulta. Recomendamos esse recurso como uma alternativa ao uso de `pg_stats_statements`.

Otimizar inserções em massa e usas dados transitórios

Se você tem operações de carga de trabalho que envolvem dados temporários ou inserem grandes conjuntos de dados em massa, considere o uso de tabelas não registradas. Por padrão, ele fornece atomicidade e durabilidade. Atomicidade, consistência, isolamento e durabilidade são as propriedades ACID. Confira [Como otimizar inserções em massa](#).

Próximas etapas

[Guia do Postgres](#)

Bibliotecas de conexão do Banco de Dados do Azure para PostgreSQL - Servidor único

09/08/2021 • 2 minutes to read

Este artigo lista as bibliotecas e os drivers que os desenvolvedores podem usar para desenvolver aplicativos para conexão e consulta do Banco de Dados do Azure para PostgreSQL.

Interfaces de cliente

A maioria das bibliotecas de cliente de linguagem usadas para se conectar ao servidor PostgreSQL são projetos externos distribuídos de forma independente. As bibliotecas listadas têm suporte em plataformas Windows, Linux e Mac para conexão ao Banco de Dados do Azure para PostgreSQL. Vários exemplos de início rápido são listados na seção Próximas etapas.

LINGUAGEM	INTERFACE DO CLIENTE	INFORMAÇÕES ADICIONAIS	DOWNLOAD
Python	psycopg	Em conformidade com a DB API 2.0	Download
PHP	php-pgsql	Extensão de banco de dados	Instalar
Node.js	Pg npm package	Cliente puro do JavaScript sem bloqueio	Instalar
Java	JDBC	Driver JDBC do tipo 4	Download
Ruby	Pg gem	Interface Ruby	Download
Go	Package pq	Driver de postgres Go puro	Instalar
C#/ .NET	Npgsql	provedor de dados ADO.NET	Download
ODBC	pgsqlODBC	Driver ODBC	Download
C	libpq	Interface primária de linguagem C	Incluso
C++	libpqxx	Interface de C++ com novo estilo	Download

Próximas etapas

Leia estes guias de início rápido para saber como se conectar e consultar o Banco de Dados do Azure para PostgreSQL usando a linguagem de sua escolha:

[Python](#) | [Node.js](#) | [Java](#) | [Ruby](#) | [PHP](#) | [.NET \(C#\)](#) | [Go](#)

Manipulação de erros de conectividade transitória para Banco de Dados do Azure para PostgreSQL – servidor único

21/05/2021 • 3 minutes to read

Este artigo descreve como tratar erros transitórios ao se conectar ao Banco de Dados do Azure para PostgreSQL.

Erros transitórios

Um erro transitório, também conhecido como uma falha transitória, é um erro que será resolvido por si só. Geralmente, esses erros manifestam como uma conexão para o servidor de banco de dados que está sendo descartado. Além disso, as novas conexões com um servidor não podem ser abertas. Os erros transitórios podem ocorrer, por exemplo, quando ocorre uma falha de hardware ou de rede. Outro motivo pode ser uma nova versão de um serviço PaaS que está sendo distribuída. A maioria desses eventos é automaticamente mitigada pelo sistema em menos de 60 segundos. Uma prática recomendada para projetar e desenvolver aplicativos na nuvem é esperar erros transitórios. Suponha que pode acontecer em qualquer componente a qualquer momento e ter a lógica apropriada em vigor para lidar com essas situações.

Tratamento de erros transitórios

Os erros transitórios devem ser manipulados usando a lógica de repetição. Situações em que devem ser consideradas:

- Ocorre um erro quando você tentar abrir uma conexão
- Uma conexão ociosa é descartada no lado do servidor. Quando você tenta emitir um comando, ele não pode ser executado
- Uma conexão ativa que esteja executando um comando é descartada.

A primeira e a segunda ocorrência são razoavelmente diretas de lidar. Tente abrir a conexão novamente. Quando você tiver êxito, o erro transitório terá sido reduzido pelo sistema. Você pode usar seu Banco de Dados do Azure para PostgreSQL novamente. Recomendamos ter esperas antes de tentar novamente a conexão. Desista se as tentativas iniciais falharem. Dessa forma, o sistema pode usar todos os recursos disponíveis para superar a situação de erro. Um bom padrão a seguir é:

- Aguarde cinco segundos até a primeira tentativa.
- Para cada próxima repetição, a espera aumenta exponencialmente, para até 60 segundos.
- Defina um número máximo de repetições no ponto em que seu aplicativo considera que a operação falhou.

Quando uma conexão com uma transação ativa falha, é mais difícil de lidar com a recuperação corretamente. Há dois casos: se a transação era somente leitura por natureza, é seguro para reabrir a conexão e tentar a transação novamente. Se, no entanto, a transação também estava gravando no banco de dados, você deve determinar se a transação foi revertida ou se ela foi bem-sucedida antes da ocorrência do erro transitório. Nesse caso, você pode simplesmente não ter recebido a confirmação do commit do servidor de banco de dados.

Uma maneira de fazer isso, é gerar uma ID exclusiva no cliente que é usado para todas as tentativas. Você pode passar essa ID exclusiva como parte da transação para o servidor e armazená-los em uma coluna com uma restrição exclusiva. Dessa forma, com segurança, você pode repetir a transação. Ela terá êxito se a transação anterior tiver sido revertida e a ID exclusiva do cliente gerada ainda não existir no sistema. Ocorrerá uma falha indicando que uma violação de chave duplicada se a ID exclusiva foi armazenada anteriormente porque a

transação anterior foi concluída com êxito.

Quando o programa se comunica com o Banco de Dados do Azure para PostgreSQL por meio de um middleware de terceiros, pergunte ao fornecedor se o middleware contém lógica de repetição para erros transitórios.

Teste a lógica de repetição. Por exemplo, tente executar seu código durante o dimensionamento os recursos de computação do seu servidor Banco de Dados do Azure para PostgreSQL. Seu aplicativo deve lidar com o breve tempo de inatividade encontrado durante a operação sem qualquer problema.

Próximas etapas

- [Solucionar problemas de conexão ao Banco de Dados do Azure para PostgreSQL](#)

Rélicas de leitura no Banco de Dados do Azure para PostgreSQL - Servidor único

12/08/2021 • 14 minutes to read

O recurso de réplica de leitura permite replicar dados de um servidor de Banco de Dados do Azure para PostgreSQL para um servidor somente leitura. As réplicas são atualizadas de forma assíncrona com a tecnologia de replicação física nativa do mecanismo PostgreSQL. Você pode replicar a partir do servidor primário para até cinco réplicas.

As réplicas são novos servidores que você gerencia de modo similar ao Banco de Dados do Azure para PostgreSQL normal. Para cada réplica de leitura, você será cobrado pela computação provisionada em vCores e pelo armazenamento em GB/mês.

Saiba como [criar e gerenciar réplicas](#).

Quando usar uma réplica de leitura

O recurso de réplica de leitura ajuda a melhorar o desempenho e o dimensionamento de cargas de trabalho com uso intenso de leitura. As cargas de trabalho de leitura podem ser isoladas para as réplicas, enquanto as cargas de trabalho de gravação podem ser direcionadas para o primário. As réplicas de leitura também podem ser implantadas em uma região diferente e podem ser promovidas a servidor de leitura/gravação em caso de recuperação de desastre.

Um cenário comum é ter cargas de trabalho analíticas e de BI usando a réplica de leitura como a fonte de dados para relatório.

Como réplicas são somente leitura, elas não reduzem diretamente os encargos de capacidade de gravação no primário.

Considerações

O recurso destina-se a cenários em que o atraso é aceitável e a descarregamento de consultas. Não é destinado a cenários de replicação síncrona em que os dados de réplica devem estar atualizados. Haverá um atraso mensurável entre o primário e a réplica. Isso pode ser em minutos ou até mesmo em horas, dependendo da carga de trabalho e da latência entre o primário e a réplica. Os dados na réplica acabarão se tornando consistentes com os dados no primário. Use este recurso para cargas de trabalho que podem acomodar esse atraso.

NOTE

Para a maioria das cargas de trabalho, as réplicas de leitura oferecem atualizações quase em tempo real a partir do primário. No entanto, com cargas de trabalho primárias persistentes, pesadas e com uso intensivo de gravação, o atraso de replicação pode continuar a aumentar e nunca alcançar o primário. Isso também pode aumentar o uso de armazenamento no primário, pois os arquivos WAL não são excluídos até que sejam recebidos na réplica. Se essa situação persistir, excluir e recriar a réplica de leitura após a conclusão das cargas de trabalho com uso intensivo de gravação é a opção de trazer a réplica de volta a um bom estado em relação ao atraso. As réplicas de leitura assíncronas não são adequadas para cargas de trabalho de gravação pesadas. Ao avaliar réplicas de leitura para seu aplicativo, monitore o atraso na réplica para um ciclo de carga de trabalho completo do aplicativo em seus horários de pico e fora de pico para acessar o possível atraso e o RTO/RPO esperado em vários pontos do ciclo de carga de trabalho.

Replicação entre regiões

Você pode criar uma réplica de leitura em uma região diferente do seu servidor primário. A replicação entre regiões pode ser útil para cenários como o planejamento de recuperação de desastres ou para trazer dados mais próximos dos seus usuários.

NOTE

Os servidores de camada básica oferecem suporte apenas à replicação de mesma região.

Você pode ter um servidor primário em qualquer [região do Banco de Dados do Azure para PostgreSQL](#). Um servidor primário pode ter uma réplica em sua região emparelhada ou nas regiões de réplica universal. A figura abaixo mostra quais regiões de réplica estão disponíveis dependendo da sua região primária.

REPLICA REGION	Australia Central	Australia Central 2	Australia East	Australia Southeast	Brazil South	Canada Central	Canada East	Central India	Central US	East Asia	East US 2	East US	France Central	Japan East	Japan West	Korea Central	Korea South	North Central US	North Europe	South Africa North	South Africa West	South Central US	South India	Southeast Asia	UAE North	UK South	UK West	West Europe	West India	West US	West US 2	West Central US
MASTER REGION																																
Australia Central	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Australia Central 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Australia East	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Australia Southeast	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Brazil South	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Canada Central	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Canada East	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Central India	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Central US	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
East Asia	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
East US 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
East US	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
France Central	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Japan East	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Japan West	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Korea Central	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Korea South	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
North Central US	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
North Europe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
South Africa North	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
South Africa West	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
South Central US	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
South India	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Southeast Asia	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
UAE North	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
UK South	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
UK West	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
West Europe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
West India	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
West US	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
West US 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
West Central US	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		

Regiões de réplica universal

Você sempre pode criar uma réplica de leitura em qualquer uma das regiões a seguir, independentemente de onde seu servidor primário esteja localizado. Estas são as regiões de réplica universal:

Leste da Austrália, Sudeste da Austrália, Sul do Brasil, Canadá Central, Leste do Canadá, EUA Central, Leste da Ásia, Leste dos EUA, Leste dos EUA 2, Leste do Japão, Oeste do Japão, Coreia Central, Sul da Coreia, Centro-Norte dos EUA, Norte da Europa, Centro-Sul dos EUA, Sudeste da Ásia, Sul do Reino Unido, Oeste do Reino Unido, Oeste da Europa, Oeste dos EUA, Oeste dos EUA 2, Centro-Oeste dos EUA.

Regiões emparelhadas

Além das regiões de réplica universal, você pode criar uma réplica de leitura na região emparelhada do Azure do seu servidor primário. Se não souber o par da sua região, você pode encontrar essa informação no [artigo Regiões emparelhadas do Azure](#).

Se você estiver usando réplicas entre regiões para o planejamento de recuperação de desastres, recomendamos criar a réplica na região emparelhada, e não em uma das outras regiões. Regiões emparelhadas evitam atualizações simultâneas e priorizam isolamento físico e residência de dados.

Limitações a serem consideradas:

- Pares unidirecionais: Algumas regiões do Azure são emparelhadas em uma direção apenas. Essas regiões incluem Oeste da Índia e Sul do Brasil. Isso significa que um servidor primário no Oeste da Índia pode criar uma réplica no Sul da Índia. Entretanto, um servidor primário no Sul da Índia não pode criar uma réplica no

Oeste da Índia. Isso ocorre porque a região secundária do Oeste da Índia é o Sul da Índia, mas a região secundária do Sul da Índia não é o Oeste da Índia.

Criar uma réplica

Quando você inicia o fluxo de trabalho de criação de réplica, um servidor do Banco de Dados do Azure para PostgreSQL é criado. O novo servidor é preenchido com os dados que estavam no servidor primário. A hora de criação depende da quantidade de dados no primário e do tempo decorrido desde o último backup completo semanal. O tempo pode variar de alguns minutos a várias horas.

Cada réplica é habilitada para armazenamento de [aumento automático](#). O recurso de aumento automático permite que a réplica acompanhe os dados replicados e evita uma interrupção na replicação causada por erros de falta de armazenamento.

O recurso de réplica de leitura usa a replicação física do PostgreSQL, e não a replicação lógica. A replicação de streaming usando slots de replicação é o modo de operação padrão. Quando necessário, o envio de logs é usado para recuperar o atraso.

Saiba como [criar uma réplica de leitura no portal do Azure](#).

Se o servidor PostgreSQL de origem estiver criptografado com chaves gerenciadas pelo cliente, consulte a [documentação](#) para obter considerações adicionais.

Conectar-se a uma réplica

Quando você cria uma réplica, ela não herda as regras de firewall nem o ponto de extremidade de serviço de rede virtual do servidor primário. Essas regras precisam ser configuradas independentemente da réplica.

A réplica herda a conta do administrador do servidor primário. Todas as contas de usuário no servidor primário são replicadas para as réplicas de leitura. Você só pode se conectar a uma réplica de leitura usando as contas de usuário disponíveis no servidor primário.

Você pode se conectar à réplica usando seu nome de host e uma conta de usuário válida, assim como faria em um servidor regular do Banco de Dados do Azure para PostgreSQL. Para um servidor chamado **my replica** com o nome de usuário administrador **myadmin**, você pode se conectar à réplica usando o `psql`:

```
psql -h myreplica.postgres.database.azure.com -U myadmin@myreplica -d postgres
```

No prompt, insira a senha da conta de usuário.

Monitorar a replicação

O Banco de Dados do Azure para PostgreSQL fornece duas métricas para monitorar a replicação. As duas métricas são **Atraso máximo entre réplicas** e **Atraso da réplica**. Para saber como exibir essas métricas, consulte a seção [Monitorar uma réplica](#) do [artigo de instruções sobre réplica de leitura](#).

A métrica **Atraso máximo entre réplicas** mostra o atraso em bytes entre o primário e a réplica com o maior atraso. Essa métrica é aplicável e está disponível somente no servidor primário e estará disponível somente se pelo menos uma das réplicas de leitura estiver conectada ao primário e o primário estiver no modo de replicação de streaming. As informações de atraso não mostram detalhes quando a réplica está no processo de alcançar o primário usando os logs arquivados do primário em um modo de replicação de envio de arquivos.

A métrica **Atraso da réplica** mostra o tempo decorrido desde a última transação reproduzida. Se não houver nenhuma transação ocorrendo no servidor primário, a métrica refletirá esse tempo decorrido. Essa métrica é aplicável e está disponível somente para servidores de réplica. O Atraso da réplica é calculado a partir da exibição de `pg_stat_wal_receiver`:

```
SELECT EXTRACT (EPOCH FROM now() - pg_last_xact_replay_timestamp());
```

Defina um alerta para informá-lo quando o retardo de réplica atinge um valor que não é aceitável para sua carga de trabalho.

Para obter mais informações, consulte diretamente o servidor primário para conhecer o atraso de replicação em bytes em todas as réplicas.

NOTE

Se um servidor primário ou a réplica de leitura forem reiniciados, o tempo que leva para reiniciar e recuperar o atraso é refletido na métrica de Atraso da réplica.

Interromper a replicação/promover a réplica

Você pode interromper a replicação entre um primário e uma réplica a qualquer momento. A ação de interrupção faz com que a réplica reinicie e promove a réplica como um servidor independente e autônomo para leitura e gravação. Os dados no servidor autônomo são os dados que estavam disponíveis no servidor de réplica no momento em que a replicação foi interrompida. Todas as atualizações subsequentes no primário não são propagadas para a réplica. No entanto, o servidor de réplica pode ter logs acumulados que ainda não foram aplicados. Como parte do processo de reinicialização, a réplica aplica todos os logs pendentes antes de aceitar conexões de cliente.

Considerações

- Antes de interromper a replicação em uma réplica de leitura, verifique o atraso da replicação para garantir que a réplica tenha todos os dados de que você precisa.
- Como a réplica de leitura precisa aplicar todos os logs pendentes antes que possa se tornar um servidor autônomo, o RTO pode ser maior para cargas de trabalho pesadas de gravação quando a replicação de interrupção acontece, pois pode haver um atraso significativo na réplica. Preste atenção a isso ao planejar a promoção de uma réplica.
- O servidor de réplica promovido não pode ser transformado em uma réplica novamente.
- Se você promover uma réplica a servidor primário, não poderá estabelecer a replicação de volta para o servidor primário antigo. Se você quiser voltar para a região primária antiga, poderá estabelecer um novo servidor de réplica com um novo nome (ou) excluir o primário antigo e criar uma réplica usando o antigo nome primário.
- Se você tiver várias réplicas de leitura e promover uma delas a servidor primário, os outros servidores de réplica ainda estarão conectados ao antigo primário. Pode ser necessário recriar réplicas a partir do novo servidor promovido.

Quando você interrompe a replicação, a réplica perde todos os links para o seu primário anterior e outras réplicas.

Saiba como [interromper a replicação para uma réplica](#).

Failover para réplica

No caso de uma falha do servidor primário, **não** ocorre o failover automático para a réplica de leitura.

Como a replicação é assíncrona, pode haver um atraso considerável entre o primário e a réplica. A quantidade de atraso é influenciada por vários fatores, como o tipo de carga de trabalho em execução no servidor primário e a latência entre o servidor primário e de réplica. Em casos típicos com carga de trabalho de gravação nominal, espera-se um atraso da réplica de alguns segundos a alguns minutos. No entanto, nos casos em que o primário executa uma carga de trabalho muito pesada e com uso intensivo de gravação e a réplica não está alcançando

com rapidez suficiente, o atraso pode ser muito maior. Você pode acompanhar o atraso de replicação de cada réplica usando a métrica *Atraso da réplica*. Essa métrica mostra o tempo decorrido desde a última transação reproduzida na réplica. Recomendamos que você identifique o atraso médio, observando o atraso da réplica ao longo de um período de tempo. Você pode definir um alerta sobre o atraso da réplica, de modo que, se ficar fora de sua faixa esperada, você seja notificado a agir.

TIP

Se você fizer failover para a réplica, o atraso no momento em que você desvincular a réplica do primário indicará quantos dados foram perdidos.

Depois de decidir que deseja fazer o failover para uma réplica:

1. Pare a replicação para a réplica

Essa etapa é necessária para fazer com que o servidor de réplica se torne um servidor autônomo e seja capaz de aceitar gravações. Como parte desse processo, o servidor de réplica será reiniciado e desconectado do primário. Depois que você inicia a interrupção da replicação, o processo de back-end normalmente leva alguns minutos para aplicar os logs residuais que ainda não foram aplicados e abrir o banco de dados como um servidor de leitura e gravação. Consulte a seção [Parar replicação](#) deste artigo para entender as implicações dessa ação.

2. Direcione seu aplicativo para a réplica (antiga)

Cada servidor tem uma cadeia de conexão única. Atualize a cadeia de conexão do aplicativo para direcionar para a réplica (antiga) em vez do primário.

Depois que seu aplicativo estiver processando leituras e gravações com êxito, você concluiu o failover. A quantidade de tempo de inatividade do seu aplicativo dependerá de quando você detectar um problema e concluir as etapas 1 e 2 acima.

Recuperação de desastre

Quando há um grande evento de desastre, como falhas regionais ou no nível de zona de disponibilidade, você pode executar a operação de recuperação de desastre promovendo a réplica de leitura. No portal de interface do usuário, navegue até o servidor de réplica de leitura. Em seguida, clique na guia de replicação e interrompa a réplica para promovê-la a servidor independente. Como alternativa, você pode usar a [CLI do Azure](#) para interromper e promover o servidor de réplica.

Considerações

Esta seção resume as considerações sobre o recurso de réplica de leitura.

Pré-requisitos

As réplicas de leitura e a [decodificação lógica](#) dependem do WAL (Log de gravação antecipada) do Postgres para obter informações. Esses dois recursos precisam de diferentes níveis de registro em log do Postgres. A decodificação lógica precisa de um nível mais alto de registro em log do que as réplicas de leitura.

Para configurar o nível certo de registro em log, use o parâmetro de suporte de replicação do Azure. O suporte à replicação do Azure tem três opções de configuração:

- **Desativada** - Coloca o mínimo de informações no WAL. Essa configuração não está disponível na maioria dos servidores do Banco de Dados do Azure para PostgreSQL.
- **Réplica** - Mais detalhada do que **Desativada**. Esse é o nível mínimo de registro em log necessário para que as [réplicas de leitura](#) funcionem. Essa configuração é o padrão na maioria dos servidores.
- **Lógica** - Mais detalhada do que **Réplica**. Este é o nível mínimo de registro em log para que a decodificação lógica funcione. As réplicas de leitura também funcionam nessa configuração.

Novas réplicas

Uma réplica de leitura é criada como um novo servidor de Banco de Dados do Azure para PostgreSQL. Um servidor existente não pode se tornar uma réplica. Você não pode criar uma réplica de outra réplica de leitura.

Configuração da réplica

Uma réplica é criada usando as mesmas configurações de computação e armazenamento que o primário. Depois que uma réplica é criada, várias configurações podem ser alteradas, incluindo o período de retenção do armazenamento e do backup.

Regras de firewall, regras de rede virtual e configurações de parâmetros não são herdadas do servidor primário para a réplica quando a réplica é criada ou posteriormente.

Scaling

Dimensionar vCores ou entre Uso geral e Otimizado para memória:

- O PostgreSQL requer que a `max_connections` configuração em um servidor secundário seja [maior ou igual à configuração no primário](#), caso contrário, o secundário não será iniciado.
- No Banco de Dados do Azure para PostgreSQL, o máximo permitido de conexões para cada servidor é corrigido para a SKU de computação, já que as conexões ocupam memória. Você pode saber mais sobre o [mapeamento entre Máximo de conexões e SKUs de computação](#).
- **Escalar verticalmente:** primeiro escala verticalmente a computação de uma réplica e, em seguida, escala verticalmente o primário. Essa ordem impedirá que erros violem o requisito `max_connections`.
- **Reducir verticalmente:** primeiro reduza verticalmente a computação do primário e, em seguida, reduza verticalmente a réplica. Se você tentar dimensionar a réplica abaixo do primário, ocorrerá um erro, pois isso viola o requisito `max_connections`.

Armazenamento em escala:

- Todas as réplicas têm o aumento automático de armazenamento habilitado para evitar problemas de replicação em uma réplica de armazenamento completo. Essa configuração não pode ser desabilitada.
- Você também pode escalar verticalmente o armazenamento manualmente, como faria em qualquer outro servidor

Camada básica

Os servidores de camada básica oferecem suporte apenas à replicação de mesma região.

`max_prepared_transactions`

O [PostgreSQL requer](#) que o valor do parâmetro `max_prepared_transactions` na réplica de leitura seja maior ou igual ao valor do primário; caso contrário, a réplica não será iniciada. Se você quiser alterar `max_prepared_transactions` no primário, primeiro altere-o nas réplicas.

Réplicas paradas

Se você interromper a replicação entre um servidor primário e uma réplica de leitura, a réplica será reiniciada para aplicar a alteração. A réplica interrompida se tornará um servidor autônomo que aceita leituras e gravações. O servidor autônomo não pode se tornar uma réplica novamente.

Servidores primário e autônomo excluídos

Quando um servidor primário é excluído, todas as suas réplicas de leitura se tornam servidores autônomos. As réplicas são reiniciadas para refletir essa alteração.

Próximas etapas

- Saiba como [criar e gerenciar réplicas de leitura no portal do Azure](#).
- Saiba como [criar e gerenciar réplicas de leitura na CLI do Azure e na API REST](#).

Decodificação lógica

18/07/2021 • 4 minutes to read

A [decodificação lógica no PostgreSQL](#) permite transmitir alterações de dados para consumidores externos. A decodificação lógica é usada popularmente para cenários de streaming de eventos e de captura de dados de alteração.

A decodificação lógica usa um plug-in de saída para converter o WAL (log de gravação antecipada) do Postgres em um formato acessível. Banco de Dados do Azure para PostgreSQL fornece os plug-ins de [saída wal2json](#), [test_decoding](#) e pgoutput. pgoutput é disponibilizado pelo PostgreSQL do PostgreSQL versão 10 e acima.

Para ter uma visão geral de como funciona a decodificação lógica do Postgres, [visite nosso blog](#).

NOTE

Não há suporte para a replicação lógica usando a publicação/assinatura do PostgreSQL com Banco de Dados do Azure para PostgreSQL - Servidor Único.

Configurar seus servidores

A decodificação lógica e as [réplicas de leitura](#) dependem do WAL (Log Write-Ahead) do Postgres para obter informações. Esses dois recursos precisam de diferentes níveis de registro em log do Postgres. A decodificação lógica precisa de um nível mais alto de registro em log do que as réplicas de leitura.

Para configurar o nível correto de registro em log, use o parâmetro de suporte à replicação do Azure. O suporte à replicação do Azure tem três opções de configuração:

- **Desativado** – coloca o mínimo de informações no WAL. Essa configuração não está disponível na maioria dos servidores do Banco de Dados do Azure para PostgreSQL.
- **Réplica** – mais detalhado do que **Desativado**. É o nível mínimo de registro em log necessário para que as [réplicas de leitura](#) funcionem. Essa configuração é o padrão na maioria dos servidores.
- **Lógico** – mais detalhado do que **Réplica**. É o nível mínimo de registro em log para que a decodificação lógica funcione. As réplicas de leitura também funcionam nessa configuração.

Usando a CLI do Azure

1. Defina azure.replication_support como `logical`.

```
az postgres server configuration set --resource-group mygroup --server-name myserver --name
azure.replication_support --value logical
```

2. Reinicie o servidor para aplicar a alteração.

```
az postgres server restart --resource-group mygroup --name myserver
```

3. Se você estiver executando o Postgres 9.5 ou 9.6 e usar o acesso à rede pública, adicione a regra de firewall para incluir o endereço IP público do cliente do qual você executará a replicação lógica. O nome da regra de firewall deve **incluir _replrule**. Por exemplo, *test_replrule*. Crie uma regra de firewall no nível de servidor do PostgreSQL do Azure com o comando [az postgres server firewall-rule create](#).

Usando o Portal do Azure

1. Definir o suporte à replicação do Azure como lógico. Selecione Salvar.

The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'myserver'. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (with Connection security, Connection strings, Server parameters, and Replication selected), and Active Directory admin. The main content area is titled 'myserver | Replication' and shows 'Azure Database for PostgreSQL server'. At the top, there are buttons for '+ Add Replica', 'Delete Replica', 'Stop Replication', 'Save' (which is highlighted with a red box), and 'Discard'. Below this, under 'Azure replication support', there are tabs for 'OFF', 'REPLICA' (which is highlighted with a red box), and 'LOGICAL'. The 'Master' section lists 'myserver' with details: Name, Pricing tier (General Purpose, 2 vCore(s)), Location (East US), and Status (Available). The 'Replicas' section shows 'No results'. At the bottom, there's a 'Save' button again.

2. Reinicie o servidor para aplicar a alteração selecionando Sim.

A modal dialog box titled 'Restart Server' appears. It contains the text: 'Applying replication support changes requires a restart. Are you sure you want to change Azure replication support to LOGICAL and restart myserver?'. At the bottom, there are two buttons: 'Yes' (highlighted with a blue border) and 'No'.

3. Se você estiver executando o Postgres 9.5 ou 9.6 e usar o acesso à rede pública, adicione a regra de firewall para incluir o endereço IP público do cliente do qual você executará a replicação lógica. O nome da regra de firewall deve **incluir _replrule**. Por exemplo, *test_replrule*. Em seguida, clique em **Salvar**.

The screenshot shows the 'Firewall rules' section of the Azure portal. It displays a table with columns for 'Firewall rule name', 'Start IP', and 'End IP'. A new rule named 'test_replrule' is listed, with both the 'Start IP' and 'End IP' fields filled with a blacked-out IP address. There are also empty rows for 'Firewall rule name', 'Start IP', and 'End IP' below it.

Iniciar decodificação lógica

A decodificação lógica pode ser consumida via protocolo de streaming ou interface do SQL. Ambos os métodos usam **slots de replicação**. Um slot representa um fluxo de alterações de um banco de dados individual.

O uso de um slot de replicação requer privilégios de replicação do Postgres. Neste momento, o privilégio de replicação só está disponível para o usuário administrador do servidor.

Protocolo de streaming

O consumo de alterações usando o protocolo de streaming geralmente é preferível. Você pode criar seu próprio consumidor/conector ou usar uma ferramenta como [Debezium](#).

Visite a documentação do wal2json para obter [um exemplo usando o protocolo de streaming com pg_recvlogical](#).

Interface do SQL

No exemplo a seguir, usamos a interface do SQL com o plug-in wal2json.

1. Crie um slot.

```
SELECT * FROM pg_create_logical_replication_slot('test_slot', 'wal2json');
```

2. Emita comandos SQL. Por exemplo:

```
CREATE TABLE a_table (
    id varchar(40) NOT NULL,
    item varchar(40),
    PRIMARY KEY (id)
);

INSERT INTO a_table (id, item) VALUES ('id1', 'item1');
DELETE FROM a_table WHERE id='id1';
```

3. Consuma as alterações.

```
SELECT data FROM pg_logical_slot_get_changes('test_slot', NULL, NULL, 'pretty-print', '1');
```

A saída terá a seguinte aparência:

```
{
  "change": [
  ]
}
{
  "change": [
    {
      "kind": "insert",
      "schema": "public",
      "table": "a_table",
      "columnnames": ["id", "item"],
      "columntypes": ["character varying(40)", "character varying(40)"],
      "columnvalues": ["id1", "item1"]
    }
  ]
}
{
  "change": [
    {
      "kind": "delete",
      "schema": "public",
      "table": "a_table",
      "oldkeys": {
        "keynames": ["id"],
        "keytypes": ["character varying(40)"],
        "keyvalues": ["id1"]
      }
    }
  ]
}
```

4. Descarte o slot quando terminar de usá-lo.

```
SELECT pg_drop_replication_slot('test_slot');
```

Slots de monitoramento

Você deve monitorar a decodificação lógica. Qualquer slot de replicação não utilizado deve ser descartado. Os slots mantêm os logs WAL do Postgres e os catálogos de sistema relevantes até que as alterações tenham sido

lidas por um consumidor. Se o consumidor falhar ou não tiver sido configurado corretamente, os logs não consumidos irão compilar e preencher o armazenamento. Além disso, os logs não consumidos aumentam o risco da ID de transação wraparound. Ambas as situações podem fazer com que o servidor fique indisponível. Portanto, é essencial que os slots de replicação lógica sejam consumidos continuamente. Se um slot de replicação lógica não for mais usado, descarte-o imediatamente.

A coluna 'ativa' na exibição pg_replication_slots indicará se há um consumidor conectado a um slot.

```
SELECT * FROM pg_replication_slots;
```

Definir alertas no *Armazenamento usado* e métricas *Retardo máximo entre as réplicas* para notificá-lo quando os valores aumentarem além dos limites normais.

IMPORTANT

Você deve soltar slots de replicação não utilizados. Não fazer isso pode levar à indisponibilidade do servidor.

Como soltar um slot

Se você não estiver consumindoativamente um slot de replicação, deverá descartá-lo.

Para soltar um slot de replicação chamado `test_slot` usando SQL:

```
SELECT pg_drop_replication_slot('test_slot');
```

IMPORTANT

Se você parar de usar a decodificação lógica, altere `azure.replication_support` para `replica` ou `off`. Os detalhes WAL retidos por `logical` são mais detalhados e devem ser desabilitados quando a decodificação lógica não estiver em uso.

Próximas etapas

- Visite a documentação do Postgres [para saber mais sobre a decodificação lógica](#).
- Entre em contato com [nossa equipe](#) se você tiver dúvidas sobre a decodificação lógica.
- Saiba mais sobre [réplicas de leitura](#).

Gerenciar um servidor do Banco de Dados do Azure para PostgreSQL usando o portal do Azure

26/05/2021 • 2 minutes to read

Este artigo mostra como gerenciar os servidores de Banco de Dados do Azure para PostgreSQL. As tarefas de gerenciamento incluem o dimensionamento da computação e do armazenamento, a redefinição de senhas do administrador e a visualização de detalhes do servidor.

Entrar

Entre no [portal do Azure](#).

Criar um servidor

Veja o [início rápido](#) para saber como criar e começar a usar um servidor do Banco de Dados do Azure para PostgreSQL.

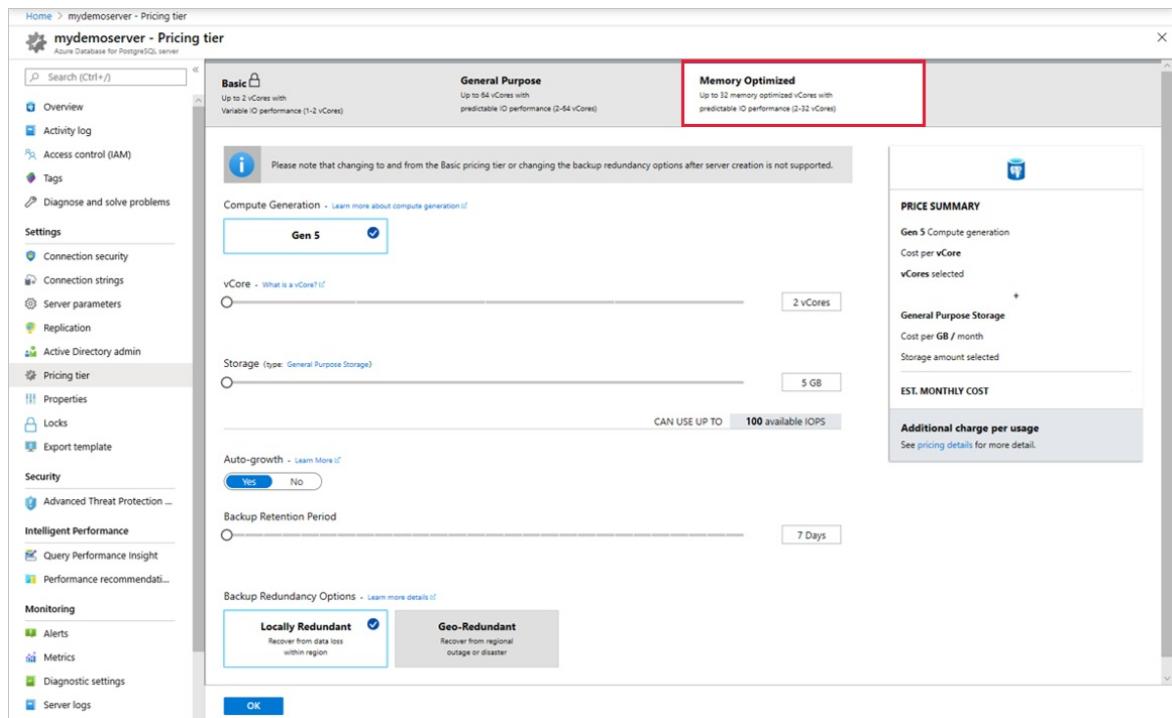
Escalar a computação e o armazenamento

Após a criação do servidor, você pode fazer o dimensionamento entre as camadas Uso Geral e Com Otimização de Memória à medida que suas necessidades mudam. Você também pode escalar o processamento e a computação aumentando ou diminuindo vCores. O armazenamento pode ser escalado verticalmente (mas não pode ser reduzido verticalmente).

Escala entre as camadas Uso Geral e Otimizado para Memória

Você pode escalar de Uso Geral para Otimizado para Memória e vice-versa. A alteração do tipo Básico após a criação do servidor não tem suporte.

1. No portal do Azure, selecione o servidor. Selecione **Tipo de preço**, localizado na seção **Configurações**.
2. Selecione **Uso Geral** ou **Otimizado para Memória**, dependendo do que você está dimensionando.



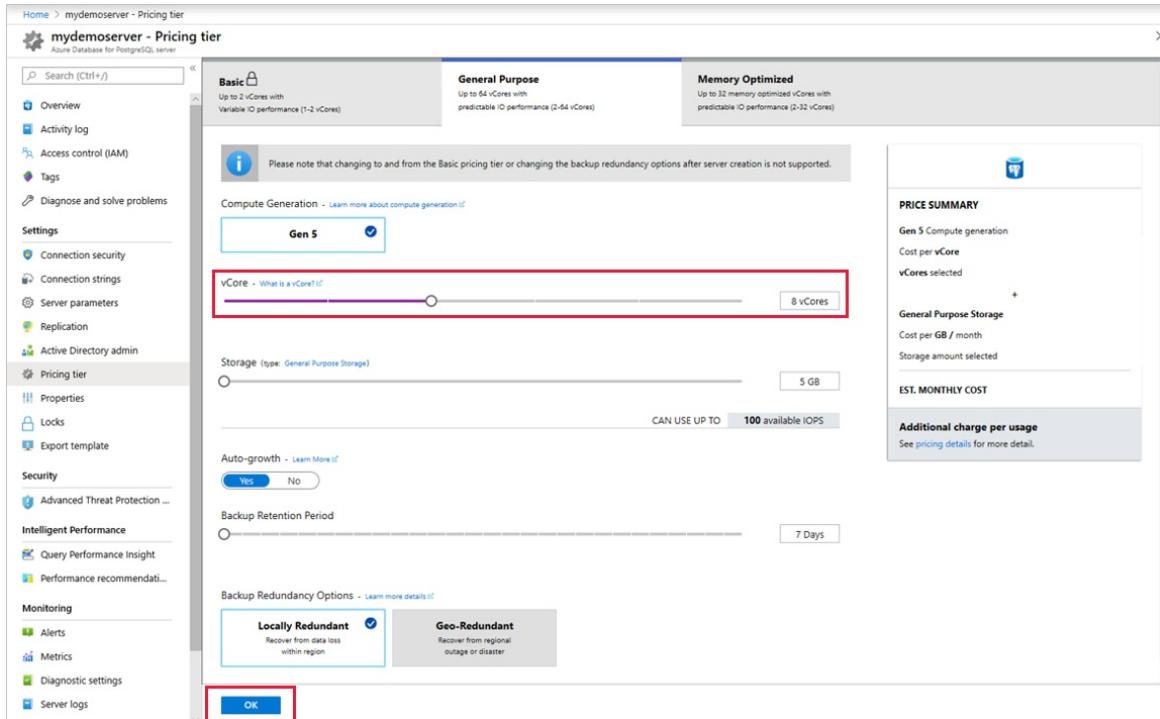
NOTE

A alteração de tipo causa uma reinicialização do servidor.

3. Selecione **OK** para salvar as alterações.

Escalar ou reduzir verticalmente vCores

1. No portal do Azure, selecione o servidor. Selecione **Tipo de preço**, localizado na seção **Configurações**.
2. Altere a configuração **vCore**, movendo o controle deslizante para o valor desejado.



NOTE

O dimensionamento de vCores causa uma reinicialização do servidor.

3. Selecione **OK** para salvar as alterações.

Escalar o armazenamento verticalmente

1. No portal do Azure, selecione o servidor. Selecione **Tipo de preço**, localizado na seção **Configurações**.
2. Altere a configuração **Armazenamento**, movendo o controle deslizante para cima até o valor desejado.

NOTE

Não é possível reduzir o armazenamento verticalmente.

3. Selecione **OK** para salvar as alterações.

Atualizar a senha do administrador

Você pode alterar a senha da função Administrador usando o portal do Azure.

1. No portal do Azure, selecione o servidor. Na janela Visão geral, selecione **Redefinir senha**.

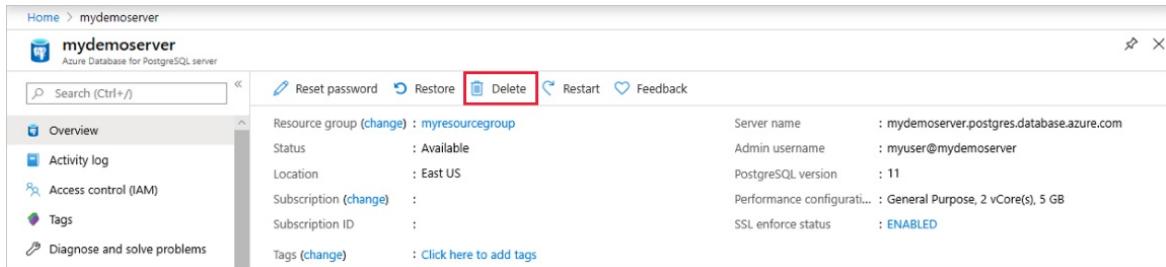
2. Insira uma nova senha e confirme-a. A caixa de texto informará os requisitos de complexidade da senha.

3. Selecione **OK** para salvar a nova senha.

Excluir um servidor

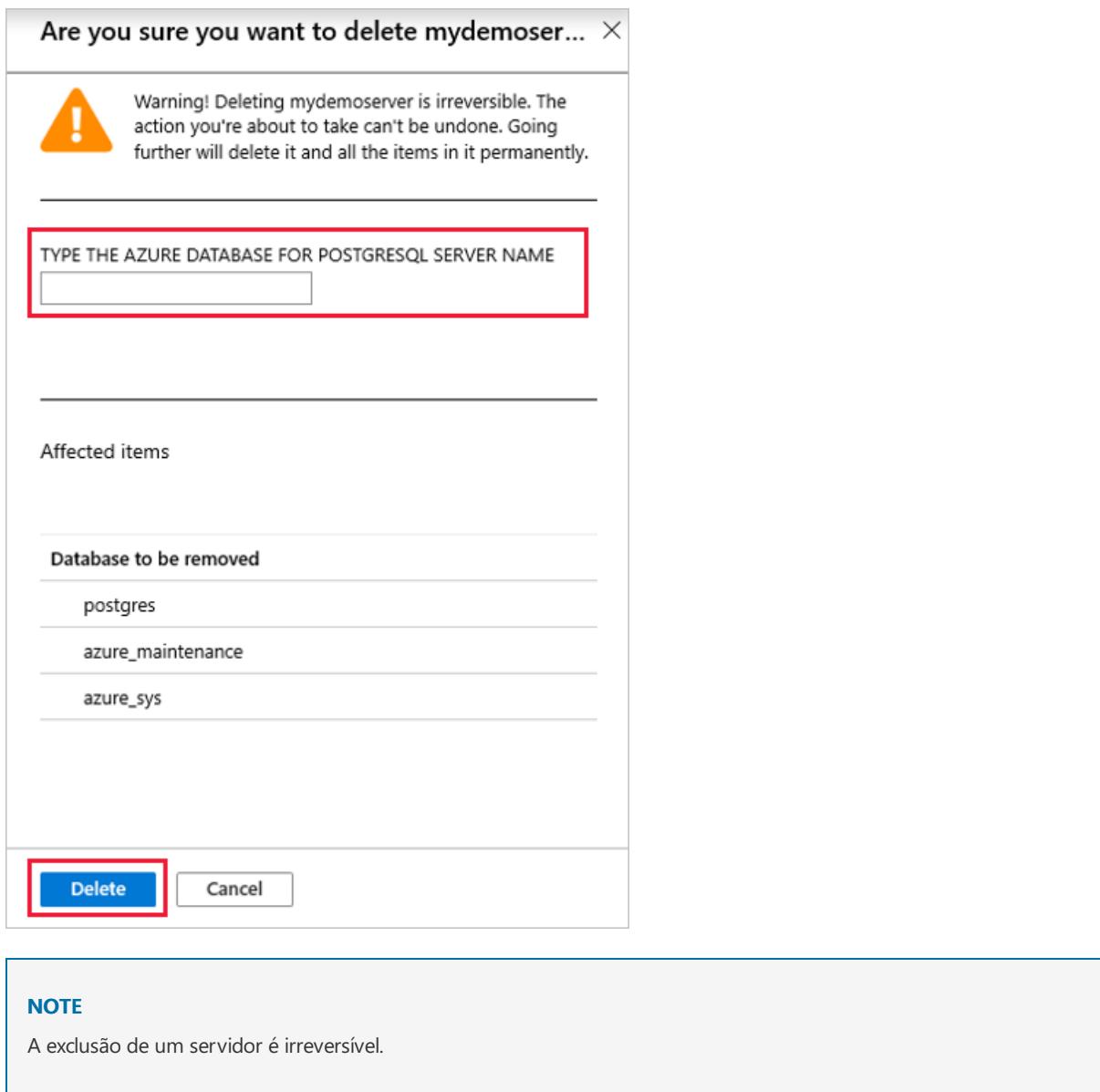
Você pode excluir o servidor se não precisar mais dele.

1. No portal do Azure, selecione o servidor. Na janela Visão geral, selecione Excluir.



The screenshot shows the Azure portal's general view for the PostgreSQL server 'mydemoserver'. The 'Delete' button in the top right corner is highlighted with a red box. The page displays various details about the server, such as its resource group, status, location, and connection information.

2. Digite o nome do servidor na caixa de entrada para confirmar que este é o servidor que você deseja excluir.



The screenshot shows a confirmation dialog box. It contains a warning message: 'Warning! Deleting mydemoserver is irreversible. The action you're about to take can't be undone. Going further will delete it and all the items in it permanently.' Below this is a text input field labeled 'TYPE THE AZURE DATABASE FOR POSTGRESQL SERVER NAME' with a red border around it. Underneath, there is a section titled 'Affected items' which lists three databases: 'postgres', 'azure_maintenance', and 'azure_sys'. At the bottom are two buttons: 'Delete' (highlighted with a red box) and 'Cancel'.

NOTE
A exclusão de um servidor é irreversível.

3. Selecione Excluir.

Próximas etapas

- Saiba mais sobre [backups e restauração do servidor](#)
- Saiba mais sobre [as opções de ajuste e monitoramento no Banco de Dados do Azure para PostgreSQL](#)

Criar um Banco de Dados do Azure para PostgreSQL - Servidor único usando a CLI do Azure

21/05/2021 • 4 minutes to read

Este artigo mostra como gerenciar seus servidores únicos implantados no Azure. As tarefas de gerenciamento incluem o dimensionamento da computação e do armazenamento, a redefinição de senhas do administrador e a visualização de detalhes do servidor.

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar. Este artigo exige que você esteja executando a CLI do Azure versão 2.0 ou posterior localmente. Para ver a versão instalada, execute o comando `az --version`. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Você precisará fazer logon em sua conta usando o comando `az login`. Observe a propriedade **id**, que se refere à **ID da Assinatura** para sua conta do Azure.

```
az login
```

Selecione a assinatura específica em sua conta usando o comando `az account set`. Anote o valor de **id** da saída `az login` para usar como valor para o argumento **subscription** no comando. Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Para obter todas as suas assinaturas, use `az account list`.

```
az account set --subscription <subscription id>
```

Caso ainda não tenha criado um servidor, crie um seguindo este guia de [início rápido](#).

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	

OPÇÃO	EXEMPLO/LINK
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

Dimensionar a computação e o armazenamento

É possível dimensionar facilmente o tipo de preço, a computação e o armazenamento seguindo o comando a seguir. Para ver todas as operações do servidor, execute [visão geral do servidor az postgres](#)

```
az postgres server update --resource-group myresourcegroup --name mydemoserver --sku-name GP_Gen5_4 --storage-size 6144
```

Estes são os detalhes dos argumentos acima:

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
name	mydemoserver	Insira um nome exclusivo que para o servidor Banco de Dados do Azure para PostgreSQL. O nome do servidor pode conter apenas letras minúsculas, números e o caractere de hífen (-). Ele deve conter de 3 a 63 caracteres.
resource-group	myresourcegroup	Forneça o nome do grupo de recursos do Azure.
sku-name	GP_Gen5_2	Insira o nome do tipo de preço e a configuração de computação. Segue a convenção {tipo de preço} {geração de computação} {vCores} em formato abreviado. Confira os tipos de preço para obter mais informações.
storage-size	6144	A capacidade de armazenamento do servidor (a unidade é megabytes). Mínimo de 5120 e aumentos em incrementos de 1024.

IMPORTANT

- O armazenamento pode ser ampliado (mas não pode ser reduzido)
- Não há suporte para dimensionar o tipo de preço Básico para Uso geral ou Com otimização de memória. Você pode realizar o dimensionamento manual [usando um script bash](#) ou [usando o PostgreSQL Workbench](#)

Gerencie bancos de dados PostgreSQL em um servidor.

Você pode usar qualquer um destes comandos para criar, excluir, listar e ver as propriedades de um banco de dados em seu servidor

CMDLET	USO	DESCRIÇÃO
az postgres db create	<pre>az postgres db create -g myresourcegroup -s mydemoserver -n mydatabasename</pre>	Cria um banco de dados
az postgres db delete	<pre>az postgres db delete -g myresourcegroup -s mydemoserver -n mydatabasename</pre>	Exclua seu banco de dados do servidor. Esse comando não exclui o servidor.
az postgres db list	<pre>az postgres db list -g myresourcegroup -s mydemoserver</pre>	Lista todos os bancos de dados no servidor
az postgres db show	<pre>az postgres db show -g myresourcegroup -s mydemoserver -n mydatabasename</pre>	Mostra mais detalhes do banco de dados

Atualizar a senha do administrador

Você pode alterar a senha da função de administrador com este comando

```
az postgres server update --resource-group myresourcegroup --name mydemoserver --admin-password <new-password>
```

IMPORTANT

Verifique se a senha tem no mínimo oito caracteres e no máximo 128. A senha deve conter caracteres de três das categorias a seguir: letras maiúsculas, letras minúsculas, números e caracteres não alfanuméricos.

Excluir um servidor

Se desejar simplesmente excluir o servidor único do PostgreSQL, será possível executar o comando [az postgres server delete](#).

```
az postgres server delete --resource-group myresourcegroup --name mydemoserver
```

Próximas etapas

- [Reiniciar um servidor](#)
- [Restaurar um servidor em um estado inadequado](#)
- [Monitorar e ajustar o servidor](#)

Reiniciar o Banco de Dados do Azure para PostgreSQL – Servidor único usando o portal do Azure

21/05/2021 • 2 minutes to read

Este tópico descreve como você pode reiniciar um servidor do Banco de Dados do Azure para PostgreSQL. Você talvez precise reiniciar o servidor por razões de manutenção, o que causa uma breve interrupção, conforme o servidor executa a operação.

A reinicialização do servidor será bloqueada se o serviço estiver ocupado. Por exemplo, o serviço pode estar processando uma operação solicitada anteriormente como o dimensionamento vCores.

NOTE

O tempo necessário para concluir uma reinicialização depende do processo de recuperação do PostgreSQL. Para diminuir o tempo de reinicialização, é recomendável que você minimize a quantidade de atividade que ocorre no servidor antes da reinicialização. Talvez você também deseje aumentar a frequência do ponto de verificação. Você pode também ajustar os valores de parâmetro relacionados ao ponto de verificação, incluindo `max_wal_size`. Também é recomendável executar o comando `CHECKPOINT` antes de reiniciar o servidor.

Pré-requisitos

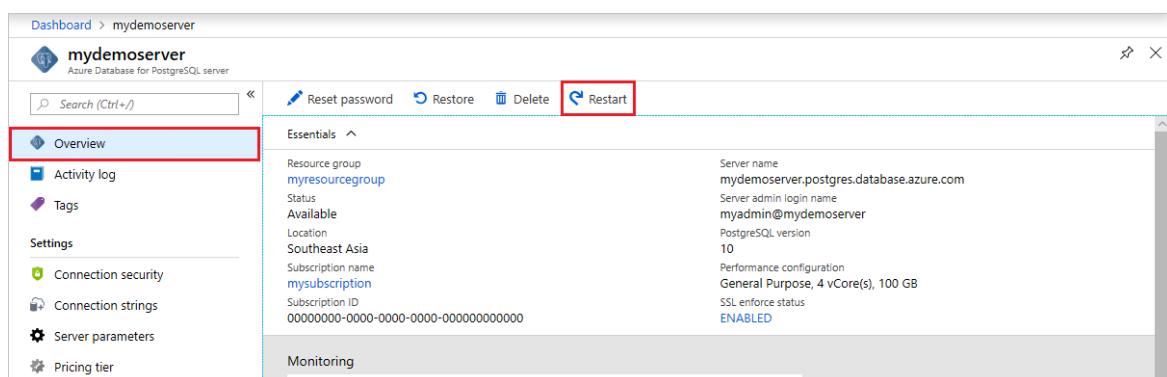
Para concluir este guia de instruções, você precisa:

- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

Realizar a reinicialização do servidor

As etapas a seguir reiniciam o servidor PostgreSQL:

1. No [portal do Azure](#), selecione o servidor do Banco de Dados do Azure para PostgreSQL.
2. Na barra de ferramentas da página de **Visão Geral** do servidor, clique em **Reiniciar**.



3. Clique em **Sim** para confirmar a reinicialização do servidor.

Dashboard > mydemoserver

mydemoserver
Azure Database for PostgreSQL server

Search (Ctrl+ /)

Overview Activity log Tags

Settings Connection security Connection strings Server parameters Pricing tier

Restart Server

Are you sure you want to restart mydemoserver?

Yes No

Southeast Asia 10 Subscription name mysubscription Performance configuration General Purpose, 4 vCore(s), 100 GB Subscription ID 00000000-0000-0000-000000000000 SSL enforce status ENABLED

Monitoring

4. Observe que o status do servidor muda para "Reiniciando".

Dashboard > mydemoserver

mydemoserver
Azure Database for PostgreSQL server

Search (Ctrl+ /)

Overview Activity log Tags

Settings Connection security Connection strings Server parameters Pricing tier

Essentials

Resource group myresourcegroup Status Restarting

Location Southeast Asia Server name mydemoserver.postgres.database.azure.com

Subscription name mysubscription Server admin login name myadmin@mydemoserver

Subscription ID 00000000-0000-0000-000000000000 PostgreSQL version 10

Performance configuration General Purpose, 4 vCore(s), 100 GB SSL enforce status ENABLED

Monitoring

*** Restarting PostgreSQL server 5:13 PM

Restarting the server mydemoserver

5. Confirme se a reinicialização do servidor foi bem-sucedida.

Dashboard > mydemoserver

mydemoserver
Azure Database for PostgreSQL server

Search (Ctrl+ /)

Overview Activity log Tags

Settings Connection security Connection strings Server parameters

Essentials

Resource group myresourcegroup Status Available

Location Southeast Asia Subscription name mysubscription

Subscription ID 00000000-0000-0000-000000000000 PostgreSQL version 10

Performance configuration General Purpose, 4 vCore(s), 100 GB SSL enforce status ENABLED

Notifications

More events in the activity log → Dismiss all ...

Successfully restarted PostgreSQL server

Successfully restarted the server mydemoserver a few seconds ago

Próximas etapas

Saiba mais sobre [como definir parâmetros no Banco de Dados do Azure para PostgreSQL](#)

Reiniciar um Banco de Dados do Azure para PostgreSQL – Servidor único usando a CLI do Azure

12/08/2021 • 2 minutes to read

Este tópico descreve como você pode reiniciar um servidor do Banco de Dados do Azure para PostgreSQL. Você talvez precise reiniciar o servidor por razões de manutenção, o que causa uma breve interrupção, conforme o servidor executa a operação.

A reinicialização do servidor será bloqueada se o serviço estiver ocupado. Por exemplo, o serviço pode estar processando uma operação solicitada anteriormente como o dimensionamento vCores.

NOTE

O tempo necessário para concluir uma reinicialização depende do processo de recuperação do PostgreSQL. Para diminuir o tempo de reinicialização, é recomendável que você minimize a quantidade de atividade que ocorre no servidor antes da reinicialização. Também se pode aumentar a frequência do ponto de verificação. Além disso, é possível ajustar os valores de parâmetro relacionados ao ponto de verificação, incluindo `max_wal_size`. Recomenda-se também executar o comando `CHECKPOINT` antes de reiniciar o servidor.

Pré-requisitos

Para concluir este guia de instruções:

- Crie um [servidor de Banco de Dados do Azure para PostgreSQL](#).
- Use o ambiente Bash no [Azure Cloud Shell](#).
A blue rectangular button with a white triangle icon on the left and the text "Launch Cloud Shell" in white on the right.
- Se preferir, [instale](#) a CLI do Azure para executar comandos de referência da CLI.
 - Se estiver usando uma instalação local, entre com a CLI do Azure usando o comando [az login](#). Para concluir o processo de autenticação, siga as etapas exibidas no terminal. Para mais opções de entrada, confira [Entrar com a CLI do Azure](#).
 - Quando solicitado, instale as extensões da CLI do Azure no primeiro uso. Para obter mais informações sobre extensões, confira [Usar extensões com a CLI do Azure](#).
 - Execute [az version](#) para localizar a versão e as bibliotecas dependentes que estão instaladas. Para fazer a atualização para a versão mais recente, execute [az upgrade](#).
- Este artigo exige a versão 2.0 ou posterior da CLI do Azure. Se você está usando o Azure Cloud Shell, a versão mais recente já está instalada.

Reinic peace o servidor

Reinic peace o servidor com o seguinte comando:

```
az postgres server restart --name mydemoserver --resource-group myresourcegroup
```

Próximas etapas

Saiba mais sobre [como definir parâmetros no Banco de Dados do Azure para PostgreSQL](#)

Reiniciar o servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell

09/08/2021 • 2 minutes to read

Este tópico descreve como você pode reiniciar um servidor do Banco de Dados do Azure para PostgreSQL. Pode ser necessário reiniciar o servidor para fins de manutenção, o que causa uma breve interrupção durante a operação.

Se o serviço estiver ocupado, a reinicialização do servidor será bloqueada. Por exemplo, o serviço pode estar processando uma operação solicitada anteriormente como o dimensionamento vCores.

NOTE

O tempo necessário para concluir uma reinicialização depende do processo de recuperação do PostgreSQL. Para diminuir o tempo de reinicialização, é recomendável que você minimize a quantidade de atividade que ocorre no servidor antes da reinicialização. Você pode aumentar a frequência do ponto de verificação. Também se pode ajustar os valores de parâmetro relacionados ao ponto de verificação, incluindo `max_wal_size`. Recomenda-se executar o comando `CHECKPOINT` antes de reiniciar o servidor.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Do módulo [AZ do PowerShell](#) instalado localmente ou do [Azure Cloud Shell](#) no navegador
- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Ao optar por usar o PowerShell localmente, conecte-se à sua conta do Azure usando o cmdlet [Connect-AzAccount](#).

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

Reinic peace o servidor

Reinic peace o servidor com o seguinte comando:

```
Restart-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup
```

Próximas etapas

[Criar um servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell](#)

Criar e gerenciar regras de firewall para o Banco de Dados do Azure para PostgreSQL – Servidor Único usando o portal do Azure

21/05/2021 • 3 minutes to read

As regras de firewall no nível de servidor podem ser usadas para gerenciar o acesso a um Banco de Dados SQL do Azure para servidor PostgreSQL de um endereço IP ou intervalo de endereços IP especificado.

As regras de VNet (rede virtual) também podem ser usadas para proteger o acesso ao seu servidor. Saiba mais sobre como [criar e gerenciar pontos de extremidade de serviço de Rede Virtual e regras usando o portal do Azure](#).

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Um servidor [Criar um servidor de Banco de Dados do Azure para o PostgreSQL](#)

Criar uma regra de firewall de nível de servidor no portal do Azure

1. Na página do servidor PostgreSQL, no título Configurações, clique em **Segurança de Conexão** para abrir a página Segurança de Conexão para o Banco de Dados do Azure para PostgreSQL.

The screenshot shows the 'mydemoserver - Connection security' page in the Azure portal. The left sidebar has sections for Overview, Activity log, Tags, SETTINGS (with Connection security highlighted with a red box), Connection strings, Server parameters, Pricing tier, Properties, Locks, MONITORING (Metrics, Alert rules, Server logs). The main content area has tabs for Save, Discard, and Add My IP. Under Access settings, 'Allow access to Azure services' is set to ON. Under SSL settings, 'Enforce SSL connection' is set to ENABLED. Under Firewall rules, it says 'Connections from the IPs specified below provides access to all the databases in privacyreviewpg.' A table for defining firewall rules is shown with columns for RULE NAME, START IP, and END IP, both currently empty. A note at the bottom says 'No firewall rules configured.'

2. Clique em **Adicionar Meu IP** na barra de ferramentas. Isso cria automaticamente uma regra de firewall com o endereço IP público do seu computador, como visto pelo sistema do Azure.

Access settings

Allow access to Azure services **ON**

SSL settings

Enforcing SSL connections on your server may require additional configuration to your applications connecting to the server. Click here to learn more.

Enforce SSL connection **ENABLED** **DISABLED**

Firewall rules

RULE NAME	START IP	END IP	...
ClientIPAddress_2018-0-4_13-41-28	131.107.147.228	131.107.147.228	...

3. Verifique seu endereço IP antes de salvar a configuração. Em algumas situações, o endereço IP observado pelo Portal do Azure é diferente do endereço IP usado ao acessar a Internet e os servidores do Azure. Portanto, talvez seja necessário alterar o IP inicial e o IP final para fazer a regra funcionar conforme o esperado. Use um mecanismo de pesquisa ou outra ferramenta online para verificar seu próprio endereço IP. Por exemplo, pesquise "qual é meu IP".

what is my ip

Web Images Videos Maps News

Also try: [where is my ip geolocation](#) · [my ip location](#) · [how to check router ip addre...](#)

30,200,000 RESULTS Any time

Your IP Address

131.107.147.228

4. Adicionar outros intervalos de endereço. Nas regras de firewall do Banco de Dados do Azure para PostgreSQL, é possível especificar um único endereço IP ou um intervalo de endereços. Se você desejar limitar a regra a um único endereço IP, digite o mesmo endereço no campo IP inicial e IP final. Abrir o firewall permite que os administradores, usuários e aplicativos acessem os bancos de dados no servidor PostgreSQL para o qual eles têm credenciais válidas.

5. Clique em **Salvar** na barra de ferramentas para salvar essa regra de firewall no nível de servidor. Aguarde a confirmação de que a atualização das regras de firewall foi bem-sucedida.

Conexão pelo Azure

Para permitir que aplicativos do Azure se conectem ao seu servidor do Banco de Dados do Azure para PostgreSQL, as conexões do Azure deverão estar habilitadas. Por exemplo, para hospedar um aplicativo dos Aplicativos Web do Azure, um aplicativo executando em uma VM do Azure ou se conectar a partir de um gateway de gerenciamento de dados do Azure Data Factory. Os recursos não precisam estar na mesma Rede Virtual (VNet) ou Grupo de Recursos para a regra de firewall habilitar essas conexões. Quando um aplicativo do Azure tenta se conectar ao seu servidor de banco de dados, o firewall verifica se há permissão para conexões do Azure. Há alguns métodos para habilitar esses tipos de conexões. Uma configuração de firewall com endereço inicial e final igual a 0.0.0.0 indica que essas conexões são permitidas. Como alternativa, é possível configurar a opção **Permitir o Acesso aos Serviços do Azure** para **ON** no portal no painel **Segurança da Conexão** e clicar em **Salvar**. Se a tentativa de conexão não for permitida, a solicitação não alcançará o servidor do Banco de Dados do Azure para PostgreSQL.

IMPORTANT

Esta opção configura o firewall para permitir todas as conexões do Azure, incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

Gerenciar regras de firewall existentes no nível de servidor pelo Portal do Azure

Repita as etapas para gerenciar as regras de firewall.

- Para adicionar o computador atual, clique no botão + **Adicionar Meu IP**. Clique em **Salvar** para salvar as alterações.
- Para adicionar mais endereços IP, digite o Nome da Regra, o Endereço IP Inicial e o Endereço IP Final. Clique em **Salvar** para salvar as alterações.
- Para modificar uma regra existente, clique em qualquer um dos campos na regra e modifique. Clique em **Salvar** para salvar as alterações.
- Para excluir uma regra existente, clique nas reticências [...] e clique em **Excluir** para remover a regra. Clique em **Salvar** para salvar as alterações.

Próximas etapas

- Da mesma forma, é possível gerar um script para [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#).
- Proteja ainda mais o acesso ao seu servidor [criando e gerenciando pontos de extremidade de serviço de Rede Virtual e regras usando o portal do Azure](#).
- Para obter ajuda com a conexão com um servidor do Banco de Dados do Azure para PostgreSQL, consulte [Bibliotecas de conexão para o Banco de Dados do Azure para PostgreSQL](#).

Criar e gerenciar regras de firewall no Banco de Dados do Azure para PostgreSQL – Servidor Único usando o CLI do Azure

21/05/2021 • 4 minutes to read

As regras de firewall no nível de servidor podem ser usadas para gerenciar o acesso a um Banco de Dados do Azure para servidor PostgreSQL de um endereço IP ou intervalo de endereços IP específico. Usando comandos convenientes da CLI do Azure, você pode criar, atualizar, excluir, listar e mostrar as regras de firewall para gerenciar o servidor. Para obter uma visão geral das regras de firewall do Banco de Dados do Azure para PostgreSQL, consulte [Regras de firewall do servidor de Banco de Dados do Azure para PostgreSQL](#).

As regras de VNet (rede virtual) também podem ser usadas para proteger o acesso ao seu servidor. Saiba mais sobre como [criar e gerenciar pontos de extremidade de serviço de Rede Virtual e regras usando o CLI do Azure](#).

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Instalar o utilitário de linha de comando [CLI do Azure](#) ou usar o Azure Cloud Shell no navegador.
- Um [banco de dados e servidor do Banco de Dados do Azure para PostgreSQL](#).

Configurar regras de firewall para o Banco de Dados do Azure para PostgreSQL

Os comandos de [az postgres server firewall-rule](#) são usados para configurar regras de firewall.

Listar regras de firewall

Para listar as regras de firewall existentes no servidor, execute o comando [az postgres server firewall-rule list](#).

```
az postgres server firewall-rule list --resource-group myresourcegroup --server-name mydemoserver
```

A saída listará as regras de firewall, se houver, no formato JSON por padrão. Você pode usar a opção [--output table](#) para obter um formato de tabela mais legível como saída.

```
az postgres server firewall-rule list --resource-group myresourcegroup --server-name mydemoserver --output table
```

Criar uma regra de firewall

Crie uma regra de firewall no nível de servidor do PostgreSQL do Azure com o comando [az postgres server firewall-rule create](#).

Para permitir o acesso a um endereço IP singular, forneça o mesmo endereço em [--start-ip-address](#) e [--end-ip-address](#), substituindo o IP mostrado aqui pelo seu IP específico, como no exemplo.

```
az postgres server firewall-rule create --resource-group myresourcegroup --server-name mydemoserver --name AllowSingleIpAddress --start-ip-address 13.83.152.1 --end-ip-address 13.83.152.1
```

Para permitir que aplicativos dos endereços IP do Azure conectem-se ao seu Banco de Dados do Azure para servidor PostgreSQL, forneça o endereço IP 0.0.0.0 como IP Inicial e IP Final, como neste exemplo.

```
az postgres server firewall-rule create --resource-group myresourcegroup --server-name mydemoserver --name AllowAllAzureIps --start-ip-address 0.0.0.0 --end-ip-address 0.0.0.0
```

IMPORTANT

Esta opção configura o firewall para permitir todas as conexões do Azure, incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

Após o êxito, a saída do comando listará os detalhes da regra do firewall que você criou, em formato JSON por padrão. Em caso de falha, a saída mostrará uma mensagem de erro.

Atualizar uma regra de firewall

Atualize uma regra de firewall existente no servidor usando o comando [az postgres server firewall-rule update](#). Forneça o nome da regra de firewall existente como entrada, e os atributos IP inicial e IP final para atualizar.

```
az postgres server firewall-rule update --resource-group myresourcegroup --server-name mydemoserver --name AllowIpRange --start-ip-address 13.83.152.0 --end-ip-address 13.83.152.0
```

Após o êxito, a saída do comando listará os detalhes da regra do firewall que você atualizou, em formato JSON por padrão. Em caso de falha, a saída mostrará uma mensagem de erro.

NOTE

Se a regra de firewall não existir, ela será criada pelo comando de atualização.

Mostrar detalhes da regra de firewall

Também é possível exibir os detalhes de uma regra de firewall existente no nível do servidor executando o comando [az postgres server firewall-rule show](#).

```
az postgres server firewall-rule show --resource-group myresourcegroup --server-name mydemoserver --name AllowIpRange
```

Após o êxito, a saída do comando listará os detalhes da regra do firewall que você especificou, em formato JSON por padrão. Em caso de falha, a saída mostrará uma mensagem de erro.

Excluir regra de firewall

Para revogar o acesso para um intervalo IP para o servidor, exclua uma regra de firewall existente executando o comando [az postgres server firewall-rule delete](#). Forneça o nome da regra de firewall existente.

```
az postgres server firewall-rule delete --resource-group myresourcegroup --server-name mydemoserver --name AllowIpRange
```

Após o êxito, não haverá saída. Em caso de falha, o texto da mensagem de erro retornará.

Próximas etapas

- Da mesma forma, é possível usar um navegador da Web para [Criar e gerenciar as regras de firewall do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure](#).
- Entenda mais sobre as [Regras de firewall do servidor de Banco de Dados do Azure para PostgreSQL](#).
- Proteja ainda mais o acesso ao seu servidor criando e gerenciando pontos de extremidade de serviço de [Rede Virtual e regras usando o CLI do Azure](#).
- Para obter ajuda com a conexão com um servidor do Banco de Dados do Azure para PostgreSQL, consulte [Bibliotecas de conexão para o Banco de Dados do Azure para PostgreSQL](#).

Conectar e consultar a visão geral para o banco de dados do Azure para PostgreSQL – Servidor Único

21/05/2021 • 2 minutes to read

O documento a seguir inclui links para exemplos que mostram como se conectar e consultar o Servidor Único do Banco de Dados do Azure para PostgreSQL. Este guia também inclui as recomendações e extensões de TLS que podem ser usadas para se conectar ao servidor em idiomas com suporte abaixo.

Inícios rápidos

GUIA DE INÍCIO RÁPIDO	DESCRIÇÃO
Pgadmin	Você pode usar o pgadmin para se conectar ao servidor e simplificar a criação, a manutenção e o uso de objetos de banco de dados.
psql no Azure Cloud Shell	Este artigo mostra como executar <code>psql</code> no Azure Cloud Shell para se conectar ao servidor e, em seguida, executar instruções para consultar, inserir, atualizar e excluir dados no banco de dados. Você pode executar o <code>psql</code> se estiver instalado em seu ambiente de desenvolvimento
PostgreSQL com VS Code	A extensão de bancos de dados do Azure para VS Code (versão prévia) permite procurar e consultar o servidor PostgreSQL localmente e na nuvem usando scrapbooks com o IntelliSense avançado.
PHP	Este guia de início rápido demonstra como usar não só o PHP para criar um programa e se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.
Java	Este guia de início rápido demonstra como usar o Java para se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.
Node.js	Este guia de início rápido demonstra como usar não só o Node.js para criar um programa e se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.
.NET(C#)	Este guia de início rápido demonstra como usar não só o .NET (C#) para criar um programa em C# e se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.
Go	Este guia de início rápido demonstra como usar o Go para se conectar a um banco de dados. As instruções Transact-SQL para consultar e modificar dados também são demonstradas.

GUIA DE INÍCIO RÁPIDO	DESCRIÇÃO
Python	Este guia de início rápido demonstra como usar o Python para se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.
Ruby	Este guia de início rápido demonstra como usar não só o Ruby para criar um programa para se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.

Considerações sobre o TLS para a conectividade de banco de dados

O TLS é usado por todos os drivers que a Microsoft fornece ou dá suporte para conexão a bancos de dados no Banco de Dados do Azure para PostgreSQL. Nenhuma configuração especial é necessária, mas imponha o TLS 1.2 para servidores recém-criados. Se você estiver usando o TLS 1.0 e 1.1, recomendamos atualizar a versão do TLS para seus servidores. Confira [Como configurar o TLS](#)

Extensões do PostgreSQL

O PostgreSQL fornece a capacidade de estender a funcionalidade de seu banco de dados usando as extensões. As extensões agrupam vários objetos SQL em um pacote que pode ser carregado ou removido do seu banco de dados com um comando. Depois de carregadas no banco de dados, as extensões funcionam como recursos internos.

- [Extensões do Postgres 11](#)
- [Extensões do Postgres 10](#)
- [Extensões do Postgres 9.6](#)

Para obter mais detalhes, confira [Como usar as extensões do PostgreSQL em um servidor único](#).

Próximas etapas

- [Migrar dados usando o despejo e a restauração](#)
- [Migrar dados usando a importação e a exportação](#)

Como fazer backup e restaurar um servidor no Banco de Dados do Azure para PostgreSQL - Servidor Único usando o portal do Azure

09/08/2021 • 5 minutes to read

O backup ocorre automaticamente

O backup do Banco de Dados do Azure para servidores PostgreSQL é feito periodicamente para habilitar os recursos de restauração. Com esse recurso de backup automático, você pode restaurar o servidor e todos os seus bancos de dados para um ponto anterior em um novo servidor.

Definir configuração de backup

Escolha entre configurar o servidor para backups com redundância local ou backups com redundância geográfica na criação do servidor, na janela **Tipo de Preço**.

NOTE

Depois que um servidor é criado, o tipo de redundância que ele tem, geográfica ou local, não pode ser alterado.

Ao criar um servidor por meio do portal do Azure, a janela **Tipo de Preço** é onde você seleciona backups **Com Redundância Local** ou **Com Redundância Geográfica** para o servidor. Essa janela também é onde você seleciona o **Período de Retenção de Backup**: quanto tempo (em número de dias) você deseja que os backups de servidor sejam armazenados.

Pricing tier

Basic Up to 2 vCores with Variable IO performance (1-2 vCores)	General Purpose Up to 64 vCores with predictable IO performance (2-64 vCores)	Memory Optimized Up to 32 memory optimized vCores with predictable IO performance (2-32 vCores)
---	--	--

Information: Please note that changing to and from the Basic pricing tier or changing the backup redundancy options after server creation is not supported.

Compute Generation: Gen 4 (disabled) | Gen 5 (selected)

vCore: What is a vCore? 4 vCores

Storage: (type: General Purpose Storage) 100 GB

Backup Retention Period: CAN USE UP TO 300 available IOPS | 7 Days

Backup Redundancy Options: Locally Redundant (disabled) | Geo-Redundant (selected)

PRICE SUMMARY:

- Gen 5 compute generation
- Cost per vCore: 4
- vCores selected: 4
- General Purpose Storage
- Cost per GB / month: 100
- Storage amount selected: 100
- EST. MONTHLY COST: USD

ADDITIONAL CHARGE PER USAGE: See [pricing details](#) for more detail.

OK

Para saber mais sobre como definir esses valores de durante a criação, confira o [guias de início rápido do Banco de Dados do Azure para servidor PostgreSQL](#).

O período de retenção de backup de um servidor pode ser alterado por meio das seguintes etapas:

1. Faça logon no [Portal do Azure](#).
2. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL. Essa ação abre a página **Visão geral** do runbook.
3. Selecione **Tipo de Preço** no menu, em **CONFIGURAÇÕES**. Usando o controle deslizante, você pode alterar o **Período de Retenção de Backup** entre 7 e 35 dias, conforme a sua preferência. Na captura de tela abaixo, ele foi aumentado para 34 dias.

4. Clique em OK para confirmar a alteração.

O período de retenção de backup determina até quando a restauração de pontos anteriores pode ser feita, já que ele se baseia em backups disponíveis. A Restauração pontual é descrita mais detalhadamente na seção a seguir.

Restauração em um momento determinado

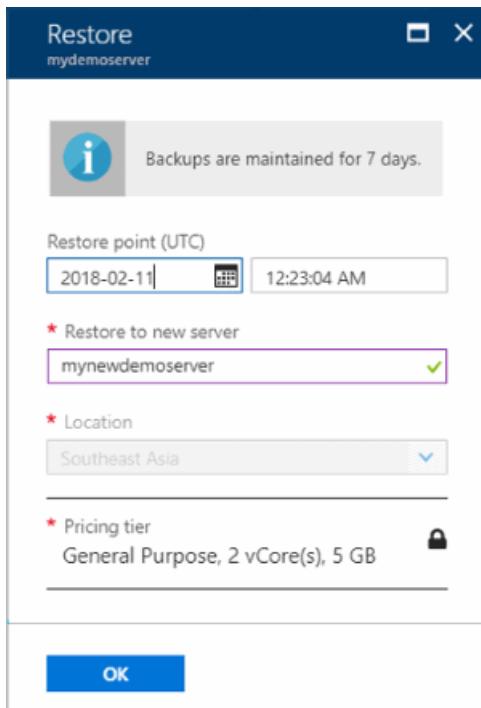
O Banco de Dados do Azure para PostgreSQL permite a restauração do servidor em um ponto anterior no tempo e em uma nova cópia do servidor. Você pode usar esse novo servidor para recuperar seus dados ou fazer seu aplicativo cliente apontar para esse novo servidor.

Por exemplo, se uma tabela for acidentalmente descartada ao meio-dia de hoje, você poderá restaurar em um momento logo antes do meio-dia e recuperar a tabela e os dados dessa nova cópia do servidor. A restauração pontual está no nível do servidor, não no nível do banco de dados.

As etapas a seguir restauraram o exemplo de servidor para um ponto anterior:

1. No Portal do Azure, selecione o servidor do Banco de Dados do Azure para PostgreSQL.
2. Na barra de ferramentas da página **Visão geral** do servidor, selecione **Restaurar**.

3. Preencha o formulário Restaurar com as informações necessárias:



- **Ponto de restauração:** selecione o ponto para o qual você deseja restaurar.
- **Servidor de destino:** forneça um nome para o novo servidor.
- **Local:** não é possível selecionar a região. Por padrão, é o mesmo que o servidor de origem.
- **Tipo de preço:** você não pode alterar esses parâmetros ao fazer uma restauração pontual. Ele é igual ao servidor de origem.

4. Clique em **OK** para restaurar o servidor em um ponto anterior.

5. Após a conclusão da restauração, localize o novo servidor criado para verificar se os dados foram restaurados conforme o esperado.

O novo servidor criado pela restauração pontual tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no ponto escolhido. Você pode alterar a senha na página **Visão geral** do novo servidor.

O novo servidor criado durante uma restauração não tem as regras de firewall ou ponto de extremidade de serviço VNet existentes no servidor original. Essas regras precisam ser configuradas separadamente para esse novo servidor.

Se o servidor PostgreSQL de origem estiver criptografado com chaves gerenciadas pelo cliente, consulte a [documentação](#) para obter considerações adicionais.

Restauração geográfica

Se você configurou seu servidor para backups com redundância geográfica, um novo servidor pode ser criado do backup do servidor existente. Esse novo servidor pode ser criado em qualquer região em que o Banco de Dados do Azure para PostgreSQL esteja disponível.

1. Selecione o botão **Criar um recurso** (+) no canto superior esquerdo do portal. Selecione **Bancos de Dados > Banco de Dados do Azure para PostgreSQL**.

The screenshot shows the Microsoft Azure portal's 'New' blade. On the left, there's a sidebar with various service icons and a 'Create a resource' button highlighted with a red box. The main area has a search bar at the top. Below it, there are tabs for 'Azure Marketplace' (selected), 'See all', and 'Featured'. The 'Featured' tab shows several database-related services like SQL Database, SQL Data Warehouse, and Azure Database for MySQL. Under the 'Azure Marketplace' tab, there's a list of categories: Get started, Recently created, Compute, Networking, Storage, Web + Mobile, Containers, Databases (which is also highlighted with a red box), Data + Analytics, AI + Cognitive Services, Internet of Things, Enterprise Integration, Security + Identity, Developer tools, Monitoring + Management, Add-ons, and Blockchain. The 'Databases' category is expanded, showing options like Azure Database for PostgreSQL (also highlighted with a red box), SQL Server 2017 Enterprise, Windows Server 2016, Azure Cosmos DB, Database as a service for MongoDB, and Redis Cache.

2. Selecione a opção de implantação Servidor único.

The screenshot shows the 'Select Azure Database for PostgreSQL deployment option' page. At the top, there's a breadcrumb trail: Home > New > Select Azure Database for PostgreSQL deployment option. The main title is 'Select Azure Database for PostgreSQL deployment option' with a note 'Microsoft - preview'. Below the title, there's a heading 'How do you plan to use the service?'. Two options are listed: 'Single server' and 'Hyperscale (Citus) server group - PREVIEW'. The 'Single server' option is highlighted with a red box. It includes a description: 'Best for broad range of traditional transactional workloads. Enterprise ready, fully managed community PostgreSQL server with up to 64 vCores, optional geospatial support, full-text search and more.', a 'Create' button, and a 'Learn more' link. The 'Hyperscale (Citus) server group - PREVIEW' option includes a description: 'Best for ultra-high performance and data needs beyond 100GB. Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response. Supports both transactional/operational workloads as well as hybrid transactional analytics workloads.', a 'Create' button, and a 'Learn more' link.

3. Forneça a assinatura, o grupo de recursos e o nome do novo servidor.

4. Selecione **Backup** como a **Fonte de dados**. Essa ação carrega um menu suspenso que fornece uma lista de servidores que têm backups com redundância geográfica habilitados.

Single server

Microsoft

X

Basics Tags Review + create

Create an Azure Database for PostgreSQL server. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

mysubscription

Resource group * ⓘ

myresourcegroup

[Create new](#)

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name * ⓘ

mygeorestoredserver

Data source * ⓘ

None **Backup**

This option allows you to restore from the most recent geo-redundant backup of any server in this subscription. The storage capacity of the server will be determined by the backup. Select a backup to continue. [Learn more](#)

Backup * ⓘ

Select a backup

Location * ⓘ

(Asia Pacific) Southeast Asia

[Supported Locations](#)

Version ⓘ

10

Compute + storage ⓘ

General Purpose

4 vCores, 100 GB storage

[Configure server](#)

[Review + create](#)

[Next : Tags >](#)

NOTE

Quando um servidor é criado pela primeira vez, talvez não fique imediatamente disponível para restauração geográfica. Pode demorar algumas horas para que os metadados necessários sejam preenchidos.

5. Selecione o menu suspenso **Backup**.

Single server

Microsoft

X

[Basics](#) [Tags](#) [Review + create](#)[Create an Azure Database for PostgreSQL server.](#) [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

mysubscription

Resource group * ⓘ

myresourcegroup

[Create new](#)

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name * ⓘ

mygeorestoredserver

Data source * ⓘ

None **Backup**

This option allows you to restore from the most recent geo-redundant backup of any server in this subscription. The storage capacity of the server will be determined by the backup. Select a backup to continue. [Learn more](#)

Backup * ⓘ

Select a backup

Location * ⓘ

(Asia Pacific) Southeast Asia

Supported Locations

Version ⓘ

10

Compute + storage ⓘ

General Purpose

4 vCores, 100 GB storage

[Configure server](#)[Review + create](#)[Next : Tags >](#)

6. Selecione o servidor de origem do qual restaurar.

Single server

Microsoft

X

[Basics](#) [Tags](#) [Review + create](#)Create an Azure Database for PostgreSQL server. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

mysubscription

Resource group * ⓘ

myresourcegroup

[Create new](#)

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name * ⓘ

mygeorestoredserver

Data source * ⓘ

None **Backup**

This option allows you to restore from the most recent geo-redundant backup of any server in this subscription. The storage capacity of the server will be determined by the backup. Select a backup to continue. [Learn more](#)

Backup * ⓘ

Select a backup

mydemoserver

mydemoserver (eastus2, 2020-06-30 23:01:21 UTC)

Version ⓘ

10

Compute + storage ⓘ

General Purpose

4 vCores, 100 GB storage

[Configure server](#)[Review + create](#)[Next : Tags >](#)

- O servidor usará como padrão os valores de número de vCores, Período de retenção de backup, Opção de redundância de backup, Versão do mecanismo e Credenciais de administrador. Selecione **Continuar**.

Single server

Microsoft

[Basics](#) [Tags](#) [Review + create](#)Create an Azure Database for PostgreSQL server. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

mysubscription



Resource group *

myresourcegroup

[Create new](#)

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name *

mygeorestoredserver



Data source *

[None](#) [Backup](#)

This option allows you to restore from the most recent geo-redundant backup of any server in this subscription. The storage capacity of the server will be determined by the backup. Select a backup to continue. [Learn more](#)

Backup *

mydemoserver (eastus2, 2020-06-30 23:01:21 UTC)



Location *

mydemoserver

Selecting this backup will modify the "Compute + Storage", credentials & version settings.

Do you want to continue?

[Continue](#)[Cancel](#)[Review + create](#)[Next : Tags >](#)

8. Preencha o restante do formulário com suas preferências. Você pode selecionar qualquer Local.

Depois de selecionar o local, você pode selecionar **Configurar servidor** para atualizar a **Geração da computação** (se disponível na região que você escolheu), número de **vCores**, **Período de retenção de backup** e **Opção de redundância de backup**. Não há suporte para a alteração do **Tipo de Preço** (Básico, Uso Geral ou Otimizado para Memória) ou do tamanho de **Armazenamento** durante a restauração.

Single server

Microsoft

X

Basics Tags Review + create

Create an Azure Database for PostgreSQL server. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	<input type="text" value="mysubscription"/>
Resource group *	<input type="text" value="myresourcegroup"/> Create new

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name *	<input type="text" value="mygeorestoredserver"/>
Data source *	<input checked="" type="radio"/> None <input checked="" type="radio"/> Backup
Backup *	<input type="text" value="mydemoserver (eastus2, 2020-06-30 23:01:21 UTC)"/>
Location *	<input type="text" value="(Asia Pacific) Southeast Asia"/>
Supported Locations	
Version	<input type="text" value="9.6"/>
Compute + storage	General Purpose 4 vCores, 100 GB storage Configure server

[Review + create](#)[Next : Tags >](#)

9. Selecione **Revisar + criar** para revisar suas seleções.

10. Selecione **Criar** para provisionar o servidor. Esta operação pode levar alguns minutos.

O novo servidor criado pela restauração geográfica tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no momento em que a restauração foi iniciada. A senha pode ser alterada na página **Visão geral** do servidor.

O novo servidor criado durante uma restauração não tem as regras de firewall ou ponto de extremidade de serviço VNet existentes no servidor original. Essas regras precisam ser configuradas separadamente para esse novo servidor.

Se o servidor PostgreSQL de origem estiver criptografado com chaves gerenciadas pelo cliente, consulte a [documentação](#) para obter considerações adicionais.

Próximas etapas

- Saiba mais sobre os [backups](#) do serviço.
- Saiba mais sobre as opções de [continuidade dos negócios](#).

Como fazer backup e restaurar um servidor no Banco de Dados do Azure para PostgreSQL - Servidor Único usando a CLI do Azure

21/05/2021 • 6 minutes to read

O backup do Banco de Dados do Azure para servidores PostgreSQL é feito periodicamente para habilitar os recursos de restauração. Com esse recurso de backup automático, você pode restaurar o servidor e todos os seus bancos de dados para um ponto anterior em um novo servidor.

Pré-requisitos

Para concluir este guia de instruções:

- Você precisa de um [banco de dados e servidor do Banco de Dados do Azure para PostgreSQL](#).
 - Use o ambiente Bash no [Azure Cloud Shell](#).
-  [Launch Cloud Shell](#)
- Se preferir, [instale](#) a CLI do Azure para executar comandos de referência da CLI.
 - Se estiver usando uma instalação local, entre com a CLI do Azure usando o comando [az login](#). Para concluir o processo de autenticação, siga as etapas exibidas no terminal. Para mais opções de entrada, confira [Entrar com a CLI do Azure](#).
 - Quando solicitado, instale as extensões da CLI do Azure no primeiro uso. Para obter mais informações sobre extensões, confira [Usar extensões com a CLI do Azure](#).
 - Execute [az version](#) para localizar a versão e as bibliotecas dependentes que estão instaladas. Para fazer a atualização para a versão mais recente, execute [az upgrade](#).
 - Este artigo exige a versão 2.0 ou posterior da CLI do Azure. Se você está usando o Azure Cloud Shell, a versão mais recente já está instalada.

Definir configuração de backup

Escolha entre configurar o servidor para backups com redundância local ou backups com redundância geográfica na criação do servidor.

NOTE

Depois que um servidor é criado, o tipo de redundância que ele tem, geográfica ou local, não pode ser alterado.

Ao criar um servidor por meio do comando `az postgres server create`, o parâmetro `--geo-redundant-backup` decide sua opção de redundância de backup. Se `Enabled`, os backups com redundância geográfica serão feitos. Ou se `Disabled`, os backups com redundância local serão feitos.

O período de retenção de backup é definido pelo parâmetro `--backup-retention-days`.

Para saber mais sobre como definir esses valores de durante a criação, confira o [Guia de início rápido do Banco de Dados do Azure para a CLI do servidor PostgreSQL](#).

O período de retenção de backup de um servidor pode ser alterado da seguinte maneira:

```
az postgres server update --name mydemoserver --resource-group myresourcegroup --backup-retention 10
```

O exemplo anterior altera o período de retenção de backup de mydemoserver para 10 dias.

O período de retenção de backup determina até quando a restauração de pontos anteriores pode ser feita, já que ele se baseia em backups disponíveis. A restauração pontual é descrita mais detalhadamente na próxima seção.

Restauração pontual do servidor

Você pode restaurar o servidor para um ponto anterior no tempo. Os dados restaurados são copiados para um novo servidor e o servidor existente é deixado como está. Por exemplo, se uma tabela tiver sido descartada acidentalmente ao meio-dia de hoje, você poderá restaurá-la para um horário um pouco antes do meio-dia. Depois, você pode recuperar a tabela e os dados ausentes da cópia restaurada do servidor.

Para restaurar o servidor, use o comando [az postgres server restore](#) da CLI do Azure.

Executar o comando `restore`

Para restaurar o servidor, no prompt de comando da CLI do Azure, digite o seguinte comando:

```
az postgres server restore --resource-group myresourcegroup --name mydemoserver-restored --restore-point-in-time 2018-03-13T13:59:00Z --source-server mydemoserver
```

O comando `az postgres server restore` exige os seguintes parâmetros:

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
resource-group	myresourcegroup	O grupo de recursos em que o servidor de origem existe.
name	mydemoserver-restored	O nome do novo servidor que é criado pelo comando de restauração.
restore-point-in-time	2018-03-13T13:59:00Z	Selecione um ponto no tempo para o qual restaurar. Essa data e hora devem estar dentro do período de retenção de backup do servidor de origem. Use o formato ISO8601 de data e hora. Por exemplo, você pode usar seu fuso horário local, como <code>2018-03-13T05:59:00-08:00</code> . Você também pode usar o formato UTC Zulu, por exemplo, <code>2018-03-13T13:59:00Z</code> .
source-server	mydemoserver	O nome ou ID para restaurar a partir do servidor de origem.

Quando você restaura um servidor para um ponto anterior no tempo, é criado um novo servidor. O servidor original e seus bancos de dados do ponto no tempo especificado são copiados para o novo servidor.

Os valores de local e tipo de preço para o servidor restaurado permanecem iguais aos do servidor de origem.

Depois que o processo de restauração é concluído, localize o novo servidor e verifique se os dados são

restaurados como esperado. O novo servidor tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no momento em que a restauração foi iniciada. A senha pode ser alterada na página **Visão geral** do servidor.

O novo servidor criado durante uma restauração não tem as regras de firewall ou ponto de extremidade de serviço VNet existentes no servidor original. Essas regras precisam ser configuradas separadamente para esse novo servidor.

Restauração geográfica

Se você configurou seu servidor para backups com redundância geográfica, um novo servidor pode ser criado do backup do servidor existente. Esse novo servidor pode ser criado em qualquer região em que o Banco de Dados do Azure para PostgreSQL esteja disponível.

Para criar um servidor usando um backup de redundância geográfica, use o comando

```
az postgres server georestore
```

 da CLI do Azure.

NOTE

Quando um servidor é criado pela primeira vez, talvez não fique imediatamente disponível para restauração geográfica. Pode demorar algumas horas para que os metadados necessários sejam preenchidos.

Para restaurar geograficamente o servidor, no prompt de comando da CLI do Azure, digite o seguinte comando:

```
az postgres server georestore --resource-group myresourcegroup --name mydemoserver-georestored --source-server mydemoserver --location eastus --sku-name GP_Gen4_8
```

Este comando cria um novo servidor chamado *mydemoserver-georestored* no Leste dos EUA que pertencerá a *myresourcegroup*. É um servidor de Geração 4 de Uso Geral com 8 vCores. O servidor é criado a partir do backup com redundância geográfica de *mydemoserver*, que também está no grupo de recursos *myresourcegroup*.

Se você deseja criar o novo servidor em outro grupo de recursos do servidor existente, será necessário qualificar no parâmetro `--source-server` o nome do servidor como no exemplo a seguir:

```
az postgres server georestore --resource-group newresourcegroup --name mydemoserver-georestored --source-server "/subscriptions/$<subscription ID>/resourceGroups/$<resource group ID>/providers/Microsoft.DBforPostgreSQL/servers/mydemoserver" --location eastus --sku-name GP_Gen4_8
```

O comando `az postgres server georestore` exige os seguintes parâmetros:

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
resource-group	myresourcegroup	O nome do grupo de recursos a qual o novo servidor pertencerá.
name	mydemoserver-georestored	O nome do novo servidor.
source-server	mydemoserver	O nome do servidor existente cujos backups com redundância geográfica são usados.
local	eastus	A localização do novo servidor.

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
sku-name	GP_Gen4_8	Esse parâmetro define o tipo de preço, a geração de computação e o número de vCores do novo servidor. GP_Gen4_8 mapeia para um servidor de Geração 4 de Uso Geral com 8 vCores.

Ao criar um novo servidor com uma restauração geográfica, ele herda o mesmo tamanho de armazenamento e tipo de preços do servidor de origem. Esses valores não podem ser alterados durante a criação. Depois que o novo servidor é criado, seu tamanho de armazenamento pode ser expandido.

Depois que o processo de restauração é concluído, localize o novo servidor e verifique se os dados são restaurados como esperado. O novo servidor tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no momento em que a restauração foi iniciada. A senha pode ser alterada na página **Visão geral** do servidor.

O novo servidor criado durante uma restauração não tem as regras de firewall ou ponto de extremidade de serviço VNet existentes no servidor original. Essas regras precisam ser configuradas separadamente para esse novo servidor.

Próximas etapas

- Saiba mais sobre os [backups](#) do serviço
- Saiba mais sobre [rélicas](#)
- Saiba mais sobre as opções de [continuidade dos negócios](#)

Como fazer backup e restaurar um servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell

21/05/2021 • 6 minutes to read

O backup dos servidores do Banco de Dados do Azure para PostgreSQL é feito periodicamente para habilitar os recursos de restauração. Com esse recurso de backup automático, você pode restaurar o servidor e todos os seus bancos de dados para um ponto anterior em um novo servidor.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Do módulo AZ do PowerShell instalado localmente ou do [Azure Cloud Shell](#) no navegador
- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Se você optar por usar o PowerShell localmente, conecte-se à sua conta do Azure usando o cmdlet [Connect-AzAccount](#).

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

Definir configuração de backup

Ao criar o servidor, você escolhe entre configurar o servidor para backups com redundância local ou geograficamente.

NOTE

Depois de criar um servidor, o tipo de redundância que ele tem, se é geográfica ou local, não pode ser alterado.

Ao criar um servidor por meio do comando `New-AzPostgreSqlServer`, o parâmetro **GeoRedundantBackup** decide sua opção de redundância de backup. Caso **Habilitado**, serão feitos os backups com redundância geográfica. Ou se caso **Desabilitado**, serão feitos os backups com redundância local.

O período de retenção de backup é definido pelo parâmetro **BackupRetentionDay**.

Para saber mais sobre como definir esses valores durante a criação do servidor, consulte [Criar um servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell](#).

O período de retenção de backup de um servidor pode ser alterado da seguinte maneira:

```
Update-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup -BackupRetentionDay 10
```

O exemplo anterior altera o período de retenção de backup de `mydemoserver` para 10 dias.

O período de retenção de backup determina até que ponto a restauração pontual pode ser feita, já que ela se baseia em backups disponíveis. A restauração pontual é descrita mais detalhadamente na próxima seção.

Restauração pontual do servidor

Você pode restaurar o servidor para um ponto anterior no tempo. Os dados restaurados são copiados para um novo servidor e o servidor existente é deixado como está. Por exemplo, se uma tabela for removida acidentalmente, você poderá restaurar até o momento em que a remoção ocorreu. Depois, você pode recuperar a tabela e os dados ausentes da cópia restaurada do servidor.

Para restaurar o servidor, use o cmdlet `Restore-AzPostgreSqlServer` do PowerShell.

Executar o comando **restore**

Para restaurar o servidor, execute o exemplo do PowerShell a seguir.

```
$restorePointInTime = (Get-Date).AddMinutes(-10)
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |
    Restore-AzPostgreSqlServer -Name mydemoserver-restored -ResourceGroupName myresourcegroup -
        RestorePointInTime $restorePointInTime -UsePointInTimeRestore
```

O conjunto de parâmetros **PointInTimeRestore** do `Restore-AzPostgreSqlServer` cmdlet exige os seguintes parâmetros:

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
ResourceGroupName	myresourcegroup	O grupo de recursos em que o servidor de origem existe.
Nome	mydemoserver-restored	O nome do novo servidor que é criado pelo comando de restauração.
RestorePointInTime	2020-03-13T13:59:00Z	Selecione um ponto no tempo para o qual restaurar. Essa data e hora devem estar dentro do período de retenção de backup do servidor de origem. Use o formato ISO8601 de data e hora. Por exemplo, você pode usar seu fuso horário local, como 2020-03-13T05:59:00-08:00. Você também pode usar o formato UTC Zulu, por exemplo, 2018-03-13T13:59:00Z.
UsePointInTimeRestore	<code><SwitchParameter></code>	Use o modo pontual para restaurar.

Quando você restaura um servidor para um ponto anterior no tempo, é criado um servidor. O servidor original e seus bancos de dados do ponto no tempo especificado são copiados para o novo servidor.

Os valores de local e tipo de preço para o servidor restaurado permanecem iguais aos do servidor de origem.

Depois que o processo de restauração é concluído, localize o novo servidor e verifique se os dados são restaurados como esperado. O novo servidor tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no momento em que a restauração foi iniciada. A senha pode ser alterada na página **Visão geral** do servidor.

O servidor criado durante uma restauração não tem o ponto de extremidade de serviço VNet existentes no servidor original. Essas regras devem ser configuradas separadamente para o novo servidor. As regras de firewall do servidor original são restauradas.

Restauração geográfica

Se você configurou seu servidor para backups com redundância geográfica, um novo servidor pode ser criado a partir do backup do servidor existente. Esse novo servidor pode ser criado em qualquer região em que o Banco de Dados do Azure para PostgreSQL esteja disponível.

Para criar um servidor usando um backup com redundância geográfica, use o `Restore-AzPostgreSqlServer` comando com o parâmetro **UseGeoRestore**.

NOTE

Quando um servidor é criado pela primeira vez, talvez não fique imediatamente disponível para restauração geográfica. Pode demorar algumas horas para que os metadados necessários sejam preenchidos.

Para restaurar a localização reográfica do servidor execute o exemplo do PowerShell a seguir:

```
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |
    Restore-AzPostgreSqlServer -Name mydemoserver-georestored -ResourceGroupName myresourcegroup -Location
    eastus -Sku GP_Gen5_8 -UseGeoRestore
```

Este exemplo cria um novo servidor chamado **mydemoserver-georestore** na região Leste dos EUA que

pertence ao **myresourcegroup**. É um Uso geral, servidor Gen 5 com 8 vCores. O servidor é criado a partir do backup com redundância geográfica de **mydemoserver**, que também está no grupo de recursos **myresourcegroup**.

Para criar o novo servidor em um grupo de recursos diferente do servidor existente, especifique o novo nome do grupo de recursos usando o parâmetro **ResourceGroupName**, conforme mostrado no exemplo a seguir:

```
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |  
    Restore-AzPostgreSqlServer -Name mydemoserver-georestored -ResourceGroupName newresourcegroup -Location  
    eastus -Sku GP_Gen5_8 -UseGeoRestore
```

O conjunto de parâmetros **GeoRestore** do `Restore-AzPostgreSqlServer` cmdlet exige os seguintes parâmetros:

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
ResourceGroupName	myresourcegroup	O nome do grupo de recursos ao qual o novo servidor pertence.
Nome	mydemoserver-georestored	O nome do novo servidor.
Local	eastus	A localização do novo servidor.
UseGeoRestore	<code><SwitchParameter></code>	Use o modo geográfico para restaurar.

Ao criar um novo servidor usando uma restauração geográfica, ele herda o mesmo tamanho de armazenamento e tipo de preços do servidor de origem a não ser que o parâmetro **SKU** seja especificado.

Depois que o processo de restauração é concluído, localize o novo servidor e verifique se os dados são restaurados como esperado. O novo servidor tem o mesmo nome de logon e senha do administrador válidos para o servidor existente no momento em que a restauração foi iniciada. A senha pode ser alterada na página **Visão geral** do servidor.

O servidor criado durante uma restauração não tem o ponto de extremidade de serviço VNet existentes no servidor original. Essas regras devem ser configuradas separadamente para este novo servidor. As regras de firewall do servidor original são restauradas.

Próximas etapas

[Como gerar uma cadeia de conexão do Banco de Dados do Azure para PostgreSQL com o PowerShell](#)

Restaurar um servidor removido do banco de dados do Azure para PostgreSQL

09/08/2021 • 2 minutes to read

Quando um servidor é descartado, o backup do servidor de banco de dados pode ser mantido em até cinco dias no serviço. O backup do banco de dados pode ser acessado e restaurado somente da assinatura do Azure em que o servidor reside originalmente. As etapas recomendadas a seguir podem ser realizadas para recuperar um recurso do servidor PostgreSQL descartado em 5 dias a partir do momento da exclusão do servidor. As etapas recomendadas só funcionarão se o backup do servidor ainda estiver disponível e não for excluído do sistema.

Pré-requisitos

Para restaurar um servidor do Banco de Dados do Azure para PostgreSQL removido, você precisará do seguinte:

- Nome da assinatura do Azure que hospeda o servidor original
- Local em que o servidor foi criado

Etapas para restauração

1. Navegue até o [Portal do Azure](#). Selecione o serviço **Azure monitor** e, em seguida, selecione **Log de atividades**.
2. Em log de atividades, clique em **Adicionar filtro**, conforme mostrado e defina os filtros conforme o seguinte
 - **Assinatura** = sua Assinatura que hospeda o servidor excluído
 - **Tipo de Recurso** = Banco de Dados do Azure para PostgreSQL (`Microsoft.DBforPostgreSQL/servers`)
 - **Operação** = Excluir servidor PostgreSQL (`Microsoft.DBforPostgreSQL/servers/delete`)

The screenshot shows the Azure Activity Log blade. At the top, there are filter options: Management Group (None), Subscription (2 selected), Event severity (All), Timespan (Last 6 hours), Resource type (selected), and a dropdown for resource name containing 'azure database for postgresql'. Below these, there are sections for Operation name and Resource ID. The Resource ID section shows a list of resources, with one item highlighted: 'Azure Database for PostgreSQL servers (Microsoft.DBforPostgreSQL/servers)'. Other items listed are 'Azure Database for PostgreSQL flexible servers (Microsoft.DBforPostgreSQL/flexibleServers)' and 'Azure Database for PostgreSQL servers v2 (Microsoft.DBforPostgreSQL/serversv2)'.

3. Selecione o evento **Excluir servidor PostgreSQL** e, em seguida, selecione a guia **JSON**. Copie os atributos `resourceId` e `submissionTimestamp` na saída JSON. O resourceId está no seguinte formato:
`/subscriptions/ffffffff-ffff-ffff-ffff-ffffffff/resourceGroups/TargetResourceGroup/providers/Microsoft.DBforPostgreSQL/servers/deletedserver`
4. Navegue até a [Página de criação da API REST do servidor](#) PostgreSQL e selecione a guia **Experimentar** realçada em verde. Faça logon usando sua conta do Azure.
5. Forneça as Propriedades `resourceGroupName`, `serverName` (nome do servidor excluído), `subscriptionId`, com base no valor JSON do atributo resourceId capturado na etapa 3 anterior. A propriedade `api-version` é preenchida previamente e pode ser deixada como está, conforme mostrado na imagem a seguir.

Servers - Create
Service: PostgreSQL
API Version: 2017-12-01
Creates a new server, or will overwrite an existing server.

URI Parameters

Name	In	Required	Type	Description
resourceGroupName	path	True	string	The name of the resource group. The name is case insensitive. Regex pattern: ^[-\w\._]{3,}+\$
serverName	path	True	string	The name of the server.
subscriptionId	path	True	string	The ID of the target subscription.
api-version	query	True	string	The API version to use for this operation.

6. Role para baixo na seção Corpo da Solicitação e cole o seguinte substituindo o "Local do Servidor Descartado" (por exemplo, CentralUS, EastUS etc.), "submissionTimestamp" e "resourceId". Para "restorePointInTime", especifique um valor de "submissionTimestamp" menos **15 minutos** para garantir que o comando não tenha erro.

```
{
  "location": "Dropped Server Location",
  "properties":
  {
    "restorePointInTime": "submissionTimestamp - 15 minutes",
    "createMode": "PointInTimeRestore",
    "sourceServerId": "resourceId"
  }
}
```

Por exemplo, se a hora atual for 2020-11-02T23:59:59.0000000Z, recomendamos um mínimo de 15 minutos antes de restaurar o ponto de restauração no tempo 2020-11-02T23:44:59.0000000Z.

IMPORTANT

Há um limite de tempo de cinco dias após o descarte do servidor. Após cinco dias, é esperado um erro, pois o arquivo de backup não pode ser encontrado.

7. Se você vir o Código de Resposta 201 ou 202, a solicitação de restauração será enviada com êxito.

A criação do servidor pode levar tempo, dependendo do tamanho do banco de dados e dos recursos de computação provisionados no servidor original. O status da restauração pode ser monitorado do log de atividades por meio da filtragem de

- **Assinatura** = Sua assinatura
- **Tipo de Recurso** = Banco de Dados do Azure para PostgreSQL (Microsoft.DBforPostgreSQL/servers)
- **Operação** = atualizar o PostgreSQL Server Create

Próximas etapas

- Se você estiver tentando restaurar um servidor em cinco dias e ainda receber um erro depois de seguir as etapas discutidas anteriormente, abra um incidente de suporte para obter assistência. Se você estiver tentando restaurar um servidor descartado após cinco dias, será esperado um erro, pois o arquivo de backup não pode ser encontrado. Não abra um tíquete de suporte neste cenário. A equipe de suporte não poderá fornecer assistência se o backup for excluído do sistema.
- Para evitar a exclusão acidental de servidores, é altamente recomendável usar [Bloqueios de Recursos](#).

Criar usuários no Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 4 minutes to read

Este artigo descreve como você pode criar usuários em um servidor do Banco de Dados do Azure para PostgreSQL.

Se desejar saber mais sobre como criar e gerenciar usuários de assinatura do Azure e seus privilégios, você poderá ver o [artigo Azure RBAC \(controle de acesso baseado em função do Azure\)](#) ou examinar [como personalizar funções](#).

A conta do administrador do servidor

Quando foi criado o Banco de Dados do Azure para PostgreSQL, você forneceu um nome de usuário de administrador de servidor e uma senha. Para saber mais, você pode seguir o [Início Rápido](#) para ver a abordagem passo a passo. Como o nome de usuário administrador do servidor é um nome personalizado, você pode localizar o nome de usuário administrador do servidor escolhido no portal do Azure.

O banco de dados do Azure para o servidor PostgreSQL é criado com as 3 funções padrão definidas. Você pode ver essas funções executando o comando: `SELECT rolname FROM pg_roles;`

- `azure_pg_admin`
- `azure_superuser`
- seu usuário administrador do servidor

O usuário administrador do servidor é um membro da função `azure_pg_admin`. No entanto, a conta do administrador do servidor não faz parte da função `azure_superuser`. Como esse serviço é um serviço gerenciado de PaaS, somente a Microsoft faz parte da função de superusuário.

O mecanismo PostgreSQL usa privilégios para controlar o acesso a objetos de banco de dados, conforme discutido na [documentação do produto PostgreSQL](#). No Banco de Dados do Azure para PostgreSQL, o usuário administrador do servidor tem estes privilégios: LOGIN, NOSUPERUSER, INHERIT, CREATEDB, CREATEROLE, NOREPLICATION

A conta de usuário administrador do servidor pode ser usada para criar usuários adicionais e conceder a esses usuários acesso à função `azure_pg_admin`. Além disso, a conta de administrador do servidor pode ser usada para criar usuários e funções com menos privilégios que têm acesso aos banco de dados e esquemas individuais.

Como criar usuários administradores adicionais no Banco de Dados do Azure para PostgreSQL

1. Obter o nome de usuário do administrador e as informações de conexão. Para conectar o servidor de banco de dados, você precisa do nome do servidor completo e as credenciais de entrada do administrador. Você pode encontrar facilmente o nome do servidor e as informações de entrada na página **Visão Geral** ou na página **Propriedades** do servidor no portal do Azure.
2. Use a conta de administrador e a senha para se conectar ao seu servidor de banco de dados. Use sua ferramenta preferencial do cliente, como pgAdmin ou psql. Se você não tem certeza de como se conectar, veja o [início rápido](#)

3. Edite e execute o código SQL a seguir. Substitua seu novo nome de usuário pelo valor de espaço reservado < new_user > e a senha de espaço reservado com sua própria senha forte.

```
CREATE ROLE <new_user> WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION PASSWORD
'<StrongPassword!>';

GRANT azure_pg_admin TO <new_user>;
```

Como criar usuários de banco de dados no Banco de Dados do Azure para PostgreSQL

1. Obter o nome de usuário do administrador e as informações de conexão. Para conectar o servidor de banco de dados, você precisa do nome do servidor completo e as credenciais de entrada do administrador. Você pode encontrar facilmente o nome do servidor e as informações de entrada na página **Visão Geral** ou na página **Propriedades** do servidor no portal do Azure.
2. Use a conta de administrador e a senha para se conectar ao seu servidor de banco de dados. Use sua ferramenta preferencial do cliente, como pgAdmin ou psql.
3. Edite e execute o código SQL a seguir. Substitua o valor de espaço reservado <db_user> com seu novo nome de usuário pretendido e o valor de espaço reservado <newdb> com seu próprio nome de banco de dados. Substitua a senha de espaço reservado pela sua própria senha forte.

Essa sintaxe de código sql cria um novo banco de dados nomeado testdb para fins de exemplo. Em seguida, ele cria um novo usuário no serviço do PostgreSQL e concede privilégios de conexão para o novo banco de dados para o usuário.

```
CREATE DATABASE <newdb>;

CREATE ROLE <db_user> WITH LOGIN NOSUPERUSER INHERIT CREATEDB NOCREATEROLE NOREPLICATION PASSWORD
'<StrongPassword!>';

GRANT CONNECT ON DATABASE <newdb> TO <db_user>;
```

4. Usando uma conta de administrador, você precisará conceder privilégios adicionais para proteger os objetos no banco de dados. Consulte a [documentação do PostgreSQL](#) para obter mais detalhes sobre as funções de banco de dados e os privilégios. Por exemplo:

```
GRANT ALL PRIVILEGES ON DATABASE <newdb> TO <db_user>;
```

5. Faça logon no servidor, especificando o banco de dados designado, usando o novo nome de usuário e senha. Este exemplo mostra a linha de comando do psql. Com este comando, será solicitada a senha para o nome de usuário. Substitua seu próprio nome de servidor, o nome do banco de dados e o nome de usuário.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=db_user@mydemoserver --
dbname=newdb
```

Próximas etapas

Abra o firewall para os endereços IP das máquinas os novos usuários para que ele possa se conectar: [Criar e gerenciar regras de firewall para o Banco de Dados do Azure para PostgreSQL usando o portal do Azure ou CLI do Azure](#).

Para saber mais sobre gerenciamento de contas de usuário, consulte a documentação do produto do PostgreSQL para [Funções e Privilégios do Banco de Dados](#), [Sintaxe de GRANT](#) e [Privilégios](#).

Usar o Azure Active Directory para autenticação com o PostgreSQL

09/08/2021 • 10 minutes to read

Este artigo explicará as etapas de como configurar o acesso do Azure Active Directory com o Banco de Dados do Azure para PostgreSQL e como se conectar usando um token do Azure Active Directory.

Configurar o usuário administrador do Azure AD

Somente usuários administradores do Azure Active Directory podem criar/habilitar usuários para a autenticação baseada no Azure Active Directory. Recomendamos não usar o administrador do Azure Active Directory para operações de banco de dados regulares, pois ele tem permissões de usuário elevadas (por exemplo, CREATEDB).

Para definir o administrador do Azure Active Directory (você pode usar um usuário ou um grupo), siga as etapas a seguir

1. No portal do Azure, selecione a instância do Banco de Dados do Azure para PostgreSQL que você deseja habilitar para o Azure Active Directory.
2. Em Configurações, selecione Administrador do Active Directory:

The screenshot shows the Azure portal interface with the URL <https://portal.azure.com/>. The user is signed in as `user@contoso.com` from the `CONTOSO` tenant. The main navigation bar has `my-azure-postgres-db - Active Directory admin` selected. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (with Connection security, Connection strings, Server parameters, Replication, Active Directory admin selected, and Pricing tier), and Help & support. The main content area is titled "my-azure-postgres-db - Active Directory admin" and describes how Azure Active Directory authentication allows central management of identity and access to the Azure Database for PostgreSQL. It shows a table with one row: "Active Directory admin" (with a status icon) and "No Active Directory admin". There are "Set admin" and "Remove admin" buttons at the top of this section, along with a "Save" button.

3. Selecione um usuário válido do Azure Active Directory no locatário do cliente para ser administrador do Azure Active Directory.

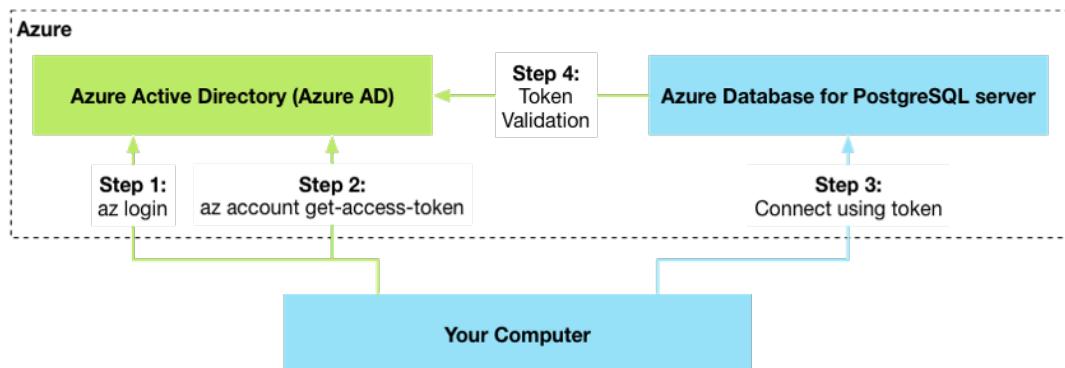
IMPORTANT

Ao definir o administrador, um novo usuário é adicionado ao servidor do Banco de Dados do Azure para PostgreSQL com permissões de administrador completo. O usuário administrador do Azure Active Directory no Banco de Dados do Azure para PostgreSQL terá a função `azure_ad_admin`. Somente um administrador do Azure Active Directory pode ser criado por servidor PostgreSQL, e a seleção de outro substituirá o administrador do Azure Active Directory existente configurado para o servidor. Você pode especificar um grupo do Azure Active Directory em vez de um usuário individual para ter vários administradores.

Somente um administrador do Azure Active Directory pode ser criado por servidor PostgreSQL, e a seleção de outro substituirá o administrador do Azure Active Directory existente configurado para o servidor. Você pode especificar um grupo do Azure Active Directory em vez de um usuário individual para ter vários administradores. Depois, tenha em mente que você entrará com o nome do grupo para fins de administração.

Conectar ao Banco de Dados do Azure para PostgreSQL usando o Azure Active Directory

O diagrama de alto nível a seguir resume o fluxo de trabalho de usar a autenticação do Azure Active Directory com o Banco de Dados do Azure para PostgreSQL:



Criamos a integração do Azure Active Directory para trabalhar com ferramentas comuns do PostgreSQL, como a `psql`, que não reconhecem o Azure Active Directory e só dão suporte à especificação de nome de usuário e senha ao se conectar ao PostgreSQL. Passamos o token do Azure Active Directory como a senha, conforme mostrado na imagem acima.

No momento, testamos os clientes a seguir:

- Linha de comando do `psql` (utilize a variável `PGPASSWORD` para transmitir o token. Confira a etapa 3 para obter mais informações)
- Azure Data Studio (usando a extensão do PostgreSQL)
- Outros clientes baseados em `libpq` (por exemplo, estruturas do aplicativo comuns e ORMs)
- PgAdmin (desmarque a opção Conectar agora durante a criação do servidor. Confira a etapa 4 para obter mais informações)

Estas são as etapas que um usuário/aplicativo precisará fazer a autenticação no Azure Active Directory descrito abaixo:

Pré-requisitos

Você pode acompanhar as etapas no Azure Cloud Shell, em uma VM do Azure ou no computador local. Garanta

que você tem o [CLI do Azure instalado](#).

Autenticar-se no Azure AD como um só usuário

Etapa 1: Fazer logon na assinatura do Azure do usuário

Comece se autenticando no Azure AD usando a ferramenta da CLI do Azure. Essa etapa não é necessária para o Azure Cloud Shell.

```
az login
```

O comando abrirá uma janela do navegador na página de autenticação do Azure AD. Ele exige que você forneça a ID de usuário e a senha do Azure Active Directory.

Etapa 2: Recuperar um token de acesso do Azure Active Directory

Invoque a ferramenta de CLI do Azure para adquirir um token de acesso para o usuário autenticado do Azure Active Directory da etapa 1 para acessar o Banco de Dados do Azure para PostgreSQL.

Exemplo (para nuvem pública):

```
az account get-access-token --resource https://osssrdbms-aad.database.windows.net
```

O valor do recurso acima deve ser especificado exatamente como o mostrado. Para outras nuvens, é possível pesquisar o valor do recurso usando:

```
az cloud show
```

Para a versão 2.0.71 do CLI do Azure e posteriores, o comando pode ser especificado na seguinte versão mais conveniente para todas as nuvens:

```
az account get-access-token --resource-type oss-rdbms
```

Depois que a autenticação for bem-sucedida, o Azure Active Directory retornará um token de acesso:

```
{
  "accessToken": "TOKEN",
  "expiresOn": "...",
  "subscription": "...",
  "tenant": "...",
  "tokenType": "Bearer"
}
```

O token é uma cadeia de caracteres Base 64 que codifica todas as informações sobre o usuário autenticado e que é direcionado para o serviço do Banco de Dados do Azure para PostgreSQL.

Etapa 3: Usar o token como senha para fazer logon com o cliente psql

Ao se conectar, você precisa usar o token de acesso como a senha de usuário do PostgreSQL.

Ao usar o cliente de linha de comando `psql`, o token de acesso precisa ser passado pela variável de ambiente `PGPASSWORD`, já que o token de acesso excede o comprimento da senha que `psql` pode aceitar diretamente:

Exemplo do Windows:

```
set PGPASSWORD=<copy/pasted TOKEN value from step 2>
```

```
$env:PGPASSWORD='<copy/pasted TOKEN value from step 2>'
```

Exemplo do Linux/macOS:

```
export PGPASSWORD=<copy/pasted TOKEN value from step 2>
```

Agora, você pode iniciar uma conexão com o Banco de Dados do Azure para PostgreSQL como faria normalmente:

```
psql "host=mydb.postgres... user=user@tenant.onmicrosoft.com@mydb dbname=postgres sslmode=require"
```

Etapa 4: Usar o token como senha para fazer logon com o PgAdmin

Para se conectar usando o token do Azure AD com o pgAdmin, siga as próximas etapas:

1. Desmarque a opção Conectar agora durante a criação do servidor.
2. Insira os detalhes do servidor na guia Conexão e salve-os.
3. No menu do navegador, clique em Conectar-se ao Banco de Dados do Azure para servidor PostgreSQL
4. Insira a senha do token do AD quando solicitado.

Considerações importantes durante a conexão:

- `user@tenant.onmicrosoft.com` é o nome do usuário do Azure AD
- Lembre-se de usar o nome do usuário do Azure exatamente como está escrito, pois os nomes de usuários e grupos do Azure AD diferenciam maiúsculas de minúsculas.
- Se o nome contiver espaços, use `\` antes de cada espaço para fazer o escape dele.
- A validade do token de acesso está entre 5 e 60 minutos. Recomendamos que você obtenha o token de acesso logo antes de iniciar o logon no Banco de Dados do Azure para PostgreSQL.

Você já está autenticado no Banco de Dados do Azure para servidor PostgreSQL usando a autenticação do Azure AD.

Autenticar-se no Azure AD como um membro do grupo

Etapa 1: Criar grupos do Azure AD no Banco de Dados do Azure para PostgreSQL

Para habilitar um grupo do Azure Active Directory para acesso ao seu banco de dados, use o mesmo mecanismo que o dos usuários, mas especifique o nome do grupo:

Exemplo:

```
CREATE ROLE "Prod DB Readonly" WITH LOGIN IN ROLE azure_ad_user;
```

Ao fazer logon, os membros do grupo usarão os tokens de acesso pessoal deles, mas assinarão com o nome do grupo especificado como o nome de usuário.

Etapa 2: Fazer logon na assinatura do Azure do usuário

Autentique-se no Azure AD usando a ferramenta da CLI do Azure. Essa etapa não é necessária para o Azure Cloud Shell. O usuário precisa ser membro do grupo do Azure AD.

```
az login
```

Etapa 3: Recuperar um token de acesso do Azure AD

Invoque a ferramenta da CLI do Azure para adquirir um token de acesso para o usuário autenticado do Azure AD na etapa 2 para acessar o Banco de Dados do Azure para PostgreSQL.

Exemplo (para nuvem pública):

```
az account get-access-token --resource https://osrdbms-aad.database.windows.net
```

O valor do recurso acima deve ser especificado exatamente como o mostrado. Para outras nuvens, é possível pesquisar o valor do recurso usando:

```
az cloud show
```

Para a versão 2.0.71 do CLI do Azure e posteriores, o comando pode ser especificado na seguinte versão mais conveniente para todas as nuvens:

```
az account get-access-token --resource-type oss-rdbms
```

Depois que a autenticação for bem-sucedida, o Azure Active Directory retornará um token de acesso:

```
{
  "accessToken": "TOKEN",
  "expiresOn": "...",
  "subscription": "...",
  "tenant": "...",
  "tokenType": "Bearer"
}
```

Etapa 4: Usar o token como senha para fazer logon com o psql ou o PgAdmin (confira as etapas acima para a conexão do usuário)

Considerações importantes durante a conexão como um membro do grupo:

- groupname@mydb é o nome do grupo do Azure AD com o qual você está tentando se conectar
- Sempre acrescente o nome do servidor após o nome de usuário/grupo do Azure AD (por exemplo, @mydb)
- Use o nome do grupo do Azure AD exatamente como está escrito.
- Os nomes de usuário e grupo do Azure AD diferenciam maiúsculas de minúsculas
- Ao se conectar como um grupo, use apenas o nome do grupo (por exemplo, GroupName@mydb), não o alias de um membro do grupo.
- Se o nome contiver espaços, use \ antes de cada espaço para fazer o escape dele.
- A validade do token de acesso está entre 5 e 60 minutos. Recomendamos que você obtenha o token de acesso logo antes de iniciar o logon no Banco de Dados do Azure para PostgreSQL.

Você já está autenticado no servidor PostgreSQL usando a autenticação do Azure Active Directory.

Criar usuários do Azure Active Directory no Banco de Dados do Azure para PostgreSQL

Para adicionar um usuário do Azure Active Directory ao banco de dados do Banco de Dados do Azure para PostgreSQL, execute as etapas a seguir após a conexão (consulte a seção mais adiante sobre como se conectar):

1. Primeiro, garanta que o usuário do Azure Active Directory `<user>@yourtenant.onmicrosoft.com` seja um usuário válido no locatário do Azure Active Directory.
2. Entre na instância do Banco de Dados do Azure para PostgreSQL como o usuário administrador do Azure Active Directory.
3. Criar a função `<user>@yourtenant.onmicrosoft.com` no Banco de Dados do Azure para PostgreSQL.
4. Torne `<user>@yourtenant.onmicrosoft.com` um membro da função `azure_ad_user`. Isso só deve ser dado aos usuários do Azure Active Directory.

Exemplo:

```
CREATE ROLE "user1@yourtenant.onmicrosoft.com" WITH LOGIN IN ROLE azure_ad_user;
```

NOTE

A autenticação de um usuário por meio do Azure Active Directory não concede ao usuário nenhuma permissão para acessar objetos no banco de dados do Banco de Dados do Azure para PostgreSQL. Você precisa conceder ao usuário as permissões necessárias manualmente.

Validação de token

A autenticação do Azure Active Directory no Banco de Dados do Azure para PostgreSQL garante que o usuário exista no servidor PostgreSQL e verifica a validade do token por meio da validação do conteúdo do token. As seguintes etapas de validação de token são executadas:

- O token é assinado pelo Azure Active Directory e não foi violado
- O token foi emitido pelo Azure Active Directory para o locatário associado ao servidor
- O token não expirou
- O token é para o recurso do Banco de Dados do Azure para PostgreSQL (e não outro recurso do Azure)

Migrando usuários existentes do PostgreSQL para a autenticação baseada no Azure Active Directory

Você pode habilitar a autenticação do Azure Active Directory para usuários existentes. Há dois casos a serem considerados:

Caso 1: O nome de usuário do PostgreSQL corresponde ao nome UPN do Azure Active Directory

No caso improvável de que os usuários existentes já correspondam aos nomes de usuário do Azure Active Directory, você pode conceder a função `azure_ad_user` a eles a fim de habilitá-los para a autenticação do Azure Active Directory:

```
GRANT azure_ad_user TO "existinguser@yourtenant.onmicrosoft.com";
```

Agora, eles poderão entrar com as credenciais do Azure Active Directory em vez de usar a senha de usuário do PostgreSQL configurada anteriormente.

Caso 2: Nome de usuário do PostgreSQL diferente do nome UPN do Azure Active Directory

Se um usuário do PostgreSQL não existir no Azure Active Directory ou tiver um nome de usuário diferente, você poderá usar os grupos do Azure Active Directory para se autenticar como este usuário do PostgreSQL. Você pode migrar os usuários existentes do Banco de Dados do Azure para PostgreSQL para o Azure Active Directory. Para isso, crie um grupo do Azure Active Directory com um nome que corresponda ao usuário do PostgreSQL e, em seguida, conceda a função `azure_ad_user` ao usuário existente do PostgreSQL:

```
GRANT azure_ad_user TO "DBReadUser";
```

Isso pressupõe que você criou um grupo "DBReadUser" no Azure Active Directory. Os usuários que pertencerem a esse grupo agora poderão entrar no banco de dados como esse usuário.

Próximas etapas

- Consulte os conceitos gerais da [Autenticação do Azure Active Directory no Banco de Dados do Azure para PostgreSQL – Servidor único](#)

Conectar com a Identidade Gerenciada ao Banco de Dados do Azure para PostgreSQL

21/05/2021 • 5 minutes to read

Este artigo mostra como usar uma identidade atribuída pelo usuário para uma VM (máquina virtual) para acessar o Banco de Dados do Azure para PostgreSQL. As Identidades de Serviço Gerenciadas são gerenciadas automaticamente pelo Azure e permitem a você autenticar os serviços que dão suporte à autenticação do Azure AD sem necessidade de inserir as credenciais em seu código.

Você aprenderá como:

- Conceder à VM acesso a um Banco de Dados do Azure para PostgreSQL
- Criar um usuário no banco de dados que represente a identidade atribuída pelo usuário da VM
- Obter um token de acesso usando a identidade da VM e usá-lo para consultar um Banco de Dados do Azure para PostgreSQL
- Implementar a recuperação de token em um aplicativo de exemplo C#

Pré-requisitos

- Se você não estiver familiarizado com as identidades gerenciadas para funcionalidades de recursos do Azure, veja esta [visão geral](#). Caso você ainda não tenha uma conta do Azure, [inscreva-se em uma conta gratuita](#) antes de continuar.
- Para executar a criação de recursos e o gerenciamento de função necessários, sua conta precisa de permissões "Proprietário" no escopo apropriado (sua assinatura ou grupo de recursos). Caso você precise de ajuda com a atribuição de função, confira [Atribuir funções do Azure para gerenciar o acesso aos recursos de assinatura do Azure](#).
- É necessário uma VM do Azure (por exemplo, ao executar Ubuntu Linux) que você gostaria de usar para acessar o banco de dados usando a Identidade Gerenciada
- Você precisa de um servidor de Banco de Dados do Azure para PostgreSQL que tenha [Autenticação do Azure AD configurada](#)
- Para seguir o exemplo C#, primeiro conclua o guia de como [Conectar com C#](#)

Criar uma identidade gerenciada atribuída pelo usuário para a VM

Crie uma identidade em sua assinatura usando o comando [az identity create](#). Você pode usar o mesmo grupo de recursos em que a máquina virtual é executada ou um diferente.

```
az identity create --resource-group myResourceGroup --name myManagedIdentity
```

Para configurar a identidade nas etapas a seguir, use o comando [az identity show](#) para armazenar a ID do recurso da entidade e a ID do cliente nas variáveis.

```

# Get resource ID of the user-assigned identity
resourceID=$(az identity show --resource-group myResourceGroup --name myManagedIdentity --query id --output tsv)

# Get client ID of the user-assigned identity
clientID=$(az identity show --resource-group myResourceGroup --name myManagedIdentity --query clientId --output tsv)

```

Agora podemos conceder a identidade atribuída pelo usuário à VM com o comando [az vm identity assign](#):

```
az vm identity assign --resource-group myResourceGroup --name myVM --identities $resourceID
```

Para concluir a instalação, mostre o valor da ID do cliente, que será necessário nas próximas etapas:

```
echo $clientID
```

Criar um usuário do PostgreSQL para sua Identidade Gerenciada

Agora, conecte-se como o usuário administrador do Azure AD ao banco de dados PostgreSQL e execute as seguintes instruções SQL:

```

SET aad_validate_oids_in_tenant = off;
CREATE ROLE myuser WITH LOGIN PASSWORD 'CLIENT_ID' IN ROLE azure_ad_user;

```

A identidade gerenciada agora tem acesso ao autenticar com o nome de usuário `myuser` (substitua pelo nome de sua escolha).

Recuperar o token de acesso do Serviço de metadados da instância do Azure

Seu aplicativo agora pode recuperar um token de acesso do Serviço de metadados da instância do Azure e usá-lo para autenticação com o banco de dados.

Essa recuperação de token é feita ao fazer uma solicitação HTTP para

`http://169.254.169.254/metadata/identity/oauth2/token` e transmitir os seguintes parâmetros:

- `api-version` = `2018-02-01`
- `resource` = `https://osssrdbms-aad.database.windows.net`
- `client_id` = `CLIENT_ID` (que você recuperou anteriormente)

Você obterá um resultado JSON que contém um campo `access_token` – esse valor de texto longo é o token de acesso da Identidade Gerenciada, que você deve usar como a senha ao se conectar ao banco de dados.

Para fins de teste, execute os seguintes comandos no shell. Observe que você precisa de `curl`, de `jq` e do cliente `psql` instalado.

```

# Retrieve the access token
export PGPASSWORD=`curl -s 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fosssrdbms-aad.database.windows.net&client_id=CLIENT_ID' -H Metadata:true | jq -r .access_token` 

# Connect to the database
psql -h SERVER --user USER@SERVER DBNAME

```

Agora você está conectado ao banco de dados configurado anteriormente.

Conectar usando a Identidade Gerenciada no C#

Esta seção mostra como obter um token de acesso usando a identidade gerenciada atribuída pelo usuário da VM e usá-lo para chamar o Banco de Dados do Azure para PostgreSQL. O Banco de Dados do Azure para PostgreSQL tem suporte nativo para autenticação do Azure AD, de modo que pode aceitar diretamente os tokens de acesso obtidos usando identidades gerenciadas para recursos do Azure. Ao criar uma conexão com o PostgreSQL, você transmite o token de acesso no campo senha.

Aqui está um exemplo de código .NET de abertura de uma conexão o PostgreSQL usando um token de acesso. Esse código deve ser executado na VM para acessar o ponto de extremidade da identidade gerenciada atribuída pelo usuário da VM. É necessário ter o .NET Framework 4.6 ou superior ou o .NET Core 2.2 ou superior para usar o método de token de acesso. Substitua os valores de HOST, USER, DATABASE e CLIENT_ID.

```
using System;
using System.Net;
using System.IO;
using System.Collections;
using System.Collections.Generic;
using System.Text.Json;
using System.Text.Json.Serialization;
using Npgsql;

namespace Driver
{
    class Script
    {
        // Obtain connection string information from the portal
        //
        private static string Host = "HOST";
        private static string User = "USER";
        private static string Database = "DATABASE";
        private static string ClientId = "CLIENT_ID";

        static void Main(string[] args)
        {
            //
            // Get an access token for PostgreSQL.
            //
            Console.Out.WriteLine("Getting access token from Azure Instance Metadata service...");

            // Azure AD resource ID for Azure Database for PostgreSQL is https://osssrbms-
            aad.database.windows.net/
            HttpWebRequest request =
            (HttpWebRequest)WebRequest.Create("http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-
            02-01&resource=https%3A%2F%2Fosssrbms-aad.database.windows.net&client_id=" + ClientId);
            request.Headers["Metadata"] = "true";
            request.Method = "GET";
            string accessToken = null;

            try
            {
                // Call managed identities for Azure resources endpoint.
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();

                // Pipe response Stream to a StreamReader and extract access token.
                StreamReader streamResponse = new StreamReader(response.GetResponseStream());
                string stringResponse = streamResponse.ReadToEnd();
                var list = JsonSerializer.Deserialize<Dictionary<string, string>>(stringResponse);
                accessToken = list["access_token"];
            }
            catch (Exception e)
            {
                Console.Out.WriteLine("SQL Connection - <Message> - <InnerException> - null");
            }
        }
    }
}
```

```

        Console.Out.WriteLine($"[{e.Message}], {e.InnerException} != null ? 
e.InnerException.Message : "Acquire token failed");
        System.Environment.Exit(1);
    }

    //
    // Open a connection to the PostgreSQL server using the access token.
    //
    string connString =
        String.Format(
            "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSLMODE=Prefer",
            Host,
            User,
            Database,
            5432,
            accessToken);

    using (var conn = new NpgsqlConnection(connString))
    {
        Console.Out.WriteLine("Opening connection using access token...");
        conn.Open();

        using (var command = new NpgsqlCommand("SELECT version()", conn))
        {

            var reader = command.ExecuteReader();
            while (reader.Read())
            {
                Console.WriteLine("\nConnected!\n\nPostgres version: {0}", reader.GetString(0));
            }
        }
    }
}

```

Quando executado, esse comando fornecerá um resultado como este:

```

Getting access token from Azure Instance Metadata service...
Opening connection using access token...

Connected!

Postgres version: PostgreSQL 11.6, compiled by Visual C++ build 1800, 64-bit

```

Próximas etapas

- Consulte os conceitos gerais da [Autenticação do Azure Active Directory com o Banco de Dados do Azure para PostgreSQL](#)

Otimizar inserções em massa e usar dados temporários em um servidor do Banco de Dados do Azure para PostgreSQL

21/05/2021 • 2 minutes to read

Este artigo descreve como você pode otimizar operações de inserção em massa e usar dados temporários em um servidor do Banco de Dados do Azure para PostgreSQL.

Usar tabelas não registradas

Se você tem operações de carga de trabalho que envolvem dados temporários ou inserem grandes conjuntos de dados em massa, considere o uso de tabelas não registradas.

Tabelas sem registro é um recurso do PostgreSQL que pode ser usado efetivamente para otimizar inserções em massa. O PostgreSQL usa WAL (registro em log write-ahead). Por padrão, ele fornece atomicidade e durabilidade. Atomicidade, consistência, isolamento e durabilidade são as propriedades ACID.

Inseri-las em uma tabela não registrada significa que o PostgreSQL faz inserções sem gravar no log de transações, que, por si só, é uma operação de E/S. Como resultado, essas tabelas são consideravelmente mais rápidas do que as tabelas comuns.

Use as seguintes opções para criar uma tabela não registrada:

- Crie uma nova tabela não registrada usando a sintaxe `CREATE UNLOGGED TABLE <tableName>`.
- Converta uma tabela registrada existente em uma tabela não registrada usando a sintaxe `ALTER TABLE <tableName> SET UNLOGGED`.

Para reverter o processo, use a sintaxe `ALTER TABLE <tableName> SET LOGGED`.

Troca de tabela sem registro

Tabelas não registradas não estão livres de falhas. Uma tabela não registrada é automaticamente truncada após uma falha ou sujeita a um desligamento não limpo. O conteúdo de uma tabela não registrada também não é replicado para servidores em espera. Quaisquer índices criados em uma tabela não registrada também são descompactados automaticamente. Após a conclusão da operação de inserção, converta a tabela para registrada para que a inserção seja durável.

Em algumas cargas de trabalho de clientes, observamos um aprimoramento no desempenho de aproximadamente 15% a 20% no uso de tabelas não registradas.

Próximas etapas

Analise sua carga de trabalho para usos de dados transitórios e inserções em massa grandes. Confira a seguinte documentação do PostgreSQL:

- [Criar comandos SQL de tabela](#)

Otimizar a aspiração automática no Banco de Dados do Azure para PostgreSQL – Servidor Único

09/08/2021 • 7 minutes to read

Este artigo descreve como otimizar de maneira eficaz o vácuo automático em um Banco de Dados do Azure para PostgreSQL.

Visão geral do vácuo automático

O PostgreSQL usa o MVCC (controle de simultaneidade multiversão) para permitir maior simultaneidade de banco de dados. Cada atualização leva a uma inserção e exclusão, e cada exclusão leva à marcação temporária de linhas para exclusão. A marcação temporária identifica tuplas inativas que serão limpas posteriormente. Para executar essas tarefas, o PostgreSQL executa um trabalho de vácuo.

Um trabalho de vácuo pode ser disparado manualmente ou automaticamente. Quando o banco de dados passa por operações com alta frequência de atualização ou exclusão, o número de tuplas inativas é maior. Quando o banco de dados está ocioso, esse número é menor. Você precisará executar trabalhos de vácuo com mais frequência quando a carga do banco de dados for pesada, o que dificulta a execução *manual* desses trabalhos.

O vácuo automático pode ser configurado e se beneficia do ajuste. Os valores padrão com que o PostgreSQL é fornecido tentam garantir que o produto funcione em todos os tipos de dispositivos. Esses dispositivos incluem o Raspberry Pis. Os valores de configuração ideais dependem:

- Do total de recursos disponíveis, como o tamanho do armazenamento e da SKU.
- Uso de recursos.
- Características de objetos individuais.

Benefícios do vácuo automático

Se você não executar o vácuo periodicamente, as tuplas inativas acumuladas poderão levar a:

- Sobrecarga de dados, como bancos de dados e tabelas maiores.
- Maiores índices de qualidade inferior.
- Aumento de E/S.

Monitorar a sobrecarga com consultas de vácuo automático

O exemplo de consulta a seguir foi criado para identificar o número de tuplas ativas e inativas em uma tabela chamada XYZ:

```
SELECT relname, n_dead_tup, n_live_tup, (n_dead_tup/ n_live_tup) AS DeadTuplesRatio, last_vacuum, last_autovacuum FROM pg_catalog.pg_stat_all_tables WHERE relname = 'XYZ' order by n_dead_tup DESC;
```

Configurações de vácuo automático

Os parâmetros de configuração que controlam o vácuo automático são baseados nas respostas de duas perguntas importantes:

- Quando ele deve começar?

- Quanto ele limpa depois de ter começado?

A seguir, há alguns parâmetros de configuração de vácuo automático que você pode atualizar com base nas perguntas acima, bem como algumas diretrizes.

PARÂMETRO	DESCRIÇÃO	VALOR PADRÃO
autovacuum_vacuum_threshold	Especifica o número mínimo de tuplas atualizadas ou excluídas necessárias para disparar uma operação de vácuo em qualquer tabela. O padrão é 50 tuplas. Este parâmetro pode ser definido apenas no arquivo postgresql.conf ou na linha de comando do servidor. Para substituir a configuração para tabelas individuais, altere os parâmetros de armazenamento da tabela.	50
autovacuum_vacuum_scale_factor	Especifica uma fração do tamanho da tabela a ser adicionada a autovacuum_vacuum_threshold ao decidir se uma operação de vácuo deve ser disparada. O padrão é 0,2, que é 20% do tamanho da tabela. Este parâmetro pode ser definido apenas no arquivo postgresql.conf ou na linha de comando do servidor. Para substituir a configuração para tabelas individuais, altere os parâmetros de armazenamento da tabela.	0,2
autovacuum_vacuum_cost_limit	Especifica o valor do limite de custo usado em operações de vácuo automáticas. Se -1, que é o padrão, for especificado, o valor normal de vacuum_cost_limit será usado. Se houver mais de uma função de trabalho, o valor será distribuído proporcionalmente entre as funções de trabalho de vácuo em execução. A soma dos limites para cada trabalho não excede o valor dessa variável. Este parâmetro pode ser definido apenas no arquivo postgresql.conf ou na linha de comando do servidor. Para substituir a configuração para tabelas individuais, altere os parâmetros de armazenamento da tabela.	-1
autovacuum_vacuum_cost_delay	Especifica o valor do atraso de custo usado em operações de vácuo automático. Se -1 for especificado, o valor normal de vacuum_cost_delay será usado. O valor padrão é 20 milissegundos. Este parâmetro pode ser definido apenas no arquivo postgresql.conf ou na linha de comando do servidor. Para substituir a configuração para tabelas individuais, altere os parâmetros de armazenamento da tabela.	20 ms

PARÂMETRO	DESCRIÇÃO	VALOR PADRÃO
autovacuum_naptime	Especifica o atraso mínimo entre execuções do vácuo automático em qualquer determinado banco de dados. Em cada turno, o daemon examina o banco de dados e emite comandos VACUUM e ANALYZE conforme necessário para as tabelas nesse banco de dados. O atraso é medido em segundos. Este parâmetro pode ser definido apenas no arquivo postgresql.conf ou na linha de comando do servidor.	15 s
autovacuum_max_workers	Especifica o número máximo de processos de vácuo automático, exceto pelo inicializador do vácuo automático, que podem ser executados simultaneamente. O padrão é três. Defina esse parâmetro somente na inicialização do servidor.	3

Para substituir as configurações de tabelas individuais, altere os parâmetros de armazenamento da tabela.

Custo do vácuo automático

A seguir, estão os "custos" de executar uma operação de vácuo:

- As páginas de dados em que o vácuo é executado são bloqueadas.
- A computação e a memória são usadas quando um trabalho de vácuo está em execução.

Sendo assim, não execute trabalhos de vácuo com frequência excessiva ou insuficiente. Um trabalho de vácuo precisa se adaptar à carga de trabalho. Teste todas as alterações de parâmetro do vácuo automático devido às vantagens e desvantagens de cada um deles.

Gatilho de início do vácuo automático

O vácuo automático é disparado quando o número de tuplas inativas ultrapassa autovacuum_vacuum_threshold + autovacuum_vacuum_scale_factor * reltuples. Aqui, reltuples é uma constante.

A limpeza promovida pelo vácuo automático deve acompanhar o ritmo do banco de dados. Caso contrário, você poderá ficar sem armazenamento e poderá ocorrer uma lentidão das consultas em geral. Amortizada ao longo do tempo, a frequência com que uma operação de vácuo limpa tuplas inativas deve ser igual à frequência com que as tuplas inativas são criadas.

Bancos de dados com muitas atualizações e exclusões têm mais tuplas inativas e requerem mais espaço. De modo geral, bancos de dados com muitas atualizações e exclusões se beneficiam de valores baixos de autovacuum_vacuum_scale_factor e autovacuum_vacuum_threshold. Os valores baixos impedem o acúmulo prolongado de tuplas inativas. Você pode usar valores mais altos para os dois parâmetros com bancos de dados menores porque a necessidade de vácuo é menos urgente. A execução frequente de operações de vácuo gera custo de computação e de memória.

O fator de escala padrão de 20 por cento funciona bem em tabelas com um percentual baixo de tuplas inativas. Por outro lado, ele não funciona bem em tabelas com alto percentual de tuplas inativas. Por exemplo, em uma tabela de 20 GB, esse fator de escala é equivalente a 4 GB de tuplas inativas. Em uma tabela de 1 TB, é equivalente a 200 GB de tuplas inativas.

Com o PostgreSQL, você pode definir esses parâmetros no nível de tabela ou no nível de instância. Atualmente, é possível definir esses parâmetros no nível da tabela apenas no Banco de Dados do Azure para PostgreSQL.

Estimar o custo do vácuo automático

Executar o vácuo automático é "caro" e há parâmetros para controlar o runtime das operações de vácuo. Os parâmetros a seguir ajudam a estimar o custo da execução de vácuo:

- `vacuum_cost_page_hit` = 1
- `vacuum_cost_page_miss` = 10
- `vacuum_cost_page_dirty` = 20

O processo de vácuo lê páginas físicas e verifica em busca de tuplas inativas. Considera-se que cada página em `shared_buffers` tem um custo igual a 1 (`vacuum_cost_page_hit`). Considera-se que todas as outras páginas têm um custo de 20 (`vacuum_cost_page_dirty`) quando há tuplas inativas ou de 10 (`vacuum_cost_page_miss`) quando não há nenhuma tupla inativa. A operação de vácuo é interrompida quando o processo ultrapassa `autovacuum_vacuum_cost_limit`.

Após o limite ser atingido, o processo fica suspenso pela duração especificada pelo parâmetro `autovacuum_vacuum_cost_delay` antes de ser iniciado novamente. Se o limite não for atingido, o vácuo automático começará após o valor especificado pelo parâmetro `autovacuum_nap_time`.

Em resumo, os parâmetros `autovacuum_vacuum_cost_delay` e `autovacuum_vacuum_cost_limit` controlam quanta limpeza de dados é permitida por unidade de tempo. Observe que os valores padrão são muito baixos para a maioria dos tipos de preço. Os valores ideais para esses parâmetros são dependentes do tipo de preço e devem ser configurados adequadamente.

O parâmetro `autovacuum_max_workers` determina o número máximo de processos de vácuo automático que podem ser executados simultaneamente.

Com o PostgreSQL, você pode definir esses parâmetros no nível de tabela ou no nível de instância. Atualmente, é possível definir esses parâmetros no nível da tabela apenas no Banco de Dados do Azure para PostgreSQL.

Otimizar o vácuo automático por tabela

Você pode configurar todos os parâmetros de configuração anteriores por tabela. Veja um exemplo:

```
ALTER TABLE t SET (autovacuum_vacuum_threshold = 1000);
ALTER TABLE t SET (autovacuum_vacuum_scale_factor = 0.1);
ALTER TABLE t SET (autovacuum_vacuum_cost_limit = 1000);
ALTER TABLE t SET (autovacuum_vacuum_cost_delay = 10);
```

O vácuo automático é um processo síncrono por tabela. Quanto maior o percentual de tuplas inativas de uma tabela, maior o "custo" do vácuo automático. Você pode dividir tabelas com alta taxa de atualizações e exclusões em várias tabelas. Dividir tabelas ajuda a paralelizar o vácuo automático e a reduzir o "custo" de concluir o vácuo automático em uma tabela. Você também pode aumentar o número de funções de trabalho de vácuo automático paralelas para garantir que os trabalhos sejam agendados livremente.

Próximas etapas

Para saber mais sobre como usar e ajustar o vácuo automático, revise a seguinte documentação do PostgreSQL:

- [Capítulo 18, Configuração do servidor](#)
- [Capítulo 24, Tarefas rotineiras de manutenção de banco de dados](#)

Otimizar a coleção de estatísticas de consulta do Banco de Dados do Azure para PostgreSQL - Servidor Único

21/05/2021 • 2 minutes to read

Este artigo descreve como otimizar a coleção de estatísticas de consulta em um servidor do Banco de Dados do Azure para PostgreSQL.

Usar pg_stats_statements

Pg_stat_statements é uma extensão do PostgreSQL habilitada por padrão no Banco de Dados do Azure para PostgreSQL. A extensão fornece um modo para rastrear as estatísticas de execução de todas as instruções SQL executadas por um servidor. Esse módulo é acoplado a cada execução de consulta e é fornecido com um custo de desempenho não trivial. Habilitar **pg_stat_statements** força as gravações de texto de consulta em arquivos no disco.

Se você tiver consultas exclusivas com texto de consulta longo ou não monitorar ativamente **pg_stat_statements**, desabilite **pg_stat_statements** para melhorar o desempenho. Para fazer isso, altere a configuração para `pg_stat_statements.track = NONE`.

Algumas cargas de trabalho de clientes têm um aumento de desempenho de até 50% quando **pg_stat_statements** é desabilitado. Em compensação, quando você desabilita **pg_stat_statements**, perde a capacidade de solucionar problemas de desempenho.

Para definir `pg_stat_statements.track = NONE`:

- No portal do Azure, vá para a [página de gerenciamento de recursos do PostgreSQL](#) e escolha a folha de [parâmetros do servidor](#).



- Use configuração de servidor az postgres da [CLI do Azure](#) definida como

```
--name pg_stat_statements.track --resource-group myresourcegroup --server mydemoserver --value NONE
```

Usar o Repositório de Consultas

O recurso [Repositório de Consultas](#) no Banco de Dados do Azure para PostgreSQL fornece um método mais eficaz para rastrear as estatísticas da consulta. Recomendamos esse recurso como uma alternativa ao uso de **pg_stats_statements**.

Próximas etapas

Considere configurar `pg_stat_statements.track = NONE` no [portal do Azure](#) ou usar a [CLI do Azure](#).

Para obter mais informações, consulte:

- [Cenários de uso do Repositório de Consultas](#)
- [Práticas recomendadas para o Repositório de Consultas](#)

Otimizar o tempo de consulta com a estratégia de armazenamento de tabela TOAST

09/08/2021 • 2 minutes to read

Este artigo descreve como otimizar os tempos de consulta com a estratégia de armazenamento de tabela TOAST (técnica de armazenamento de atributo de tamanho grande).

Estratégias de armazenamento de tabelas TOAST

Quatro estratégias diferentes são usadas para armazenar colunas em disco que podem usar o TOAST. Elas representam várias combinações entre a compactação e o armazenamento fora de linha. A estratégia pode ser definida no nível do tipo de dados e no nível da coluna.

- **Sem Formatação** impede a compactação ou o armazenamento fora de linha. Ela desabilita o uso de cabeçalhos de byte único para tipos varlena. Sem Formatação é a única estratégia possível para colunas de tipos de dados que não podem usar TOAST.
- **Estendido** permite compactação e armazenamento fora de linha. Estendido é o padrão para a maioria dos tipos de dados que podem usar TOAST. Ocorre uma tentativa de compactação primeiro. Ocorrerá uma tentativa de armazenamento fora de linha se a linha ainda for muito grande.
- **External** permite armazenamento fora de linha, mas não compactação. Uso de Externa realiza operações de subcadeia de caracteres em texto amplo e em colunas bytea mais rápido. Essa velocidade é fornecida às custas de um aumento no espaço de armazenamento. Essas operações são otimizadas para efetuar fetch apenas das partes necessárias do valor fora de linha quando ele não está compactado.
- **Principal** permite a compactação, mas não o armazenamento fora de linha. O armazenamento fora de linha ainda é realizado para essas colunas, mas apenas como último recurso. Ele ocorre quando não há outra maneira de tornar a linha pequena o suficiente para caber em uma página.

Usar estratégias de armazenamento de tabela TOAST

Se suas consultas acessarem tipos de dados que podem usar TOAST, considere usar a estratégia Principal, em vez da opção padrão Estendido para reduzir os tempos de consulta. A Principal não descarta o armazenamento fora de linha. Se suas consultas não acessam tipos de dados que podem usar TOAST, pode ser benéfico manter a opção Estendido. Uma parte maior das linhas da tabela principal cabe no cache do buffer compartilhado, que ajuda o desempenho.

Se você tiver uma carga de trabalho que usa um esquema com tabelas amplas e altas contagens de caracteres, considere usar as tabelas TOAST do PostgreSQL. Um exemplo de tabela de clientes tinha mais de 350 colunas com várias colunas que abrangiam 255 caracteres. Após sua conversão na estratégia Principal da tabela TOAST, o tempo de consulta de benchmarks foi reduzido de 4203 segundos para 467 segundos. É uma melhoria de 89%.

Próximas etapas

Analise sua carga de trabalho quanto às características anteriores.

Examine a seguinte documentação do PostgreSQL:

- [Capítulo 68, Armazenamento físico de banco de dados](#)

Definir as configurações TLS no Banco de Dados do Azure para PostgreSQL – Servidor único usando o portal do Azure

21/05/2021 • 2 minutes to read

Este artigo descreve como você pode configurar um Banco de Dados do Azure para PostgreSQL para impor a versão mínima permitida do TLS para conexões e negar todas as conexões com uma versão do TLS mais baixa do que a versão mínima do TLS, melhorando assim a segurança da rede.

Você pode impor a versão do TLS para se conectar ao Banco de Dados do Azure para PostgreSQL. Agora, os clientes têm a opção de definir a versão mínima do TLS para o servidor de banco de dados. Por exemplo, definir a versão mínima da configuração TLS como 1.0 significa que o servidor permitirá conexões de clientes usando o TLS 1.0, 1.1, 1.2 e superior. Em vez disso, definir a versão mínima do TLS como 1.2 e superior significa que você só permitirá conexões de clientes que usam o TLS 1.2 e superior e todas as conexões com o TLS 1.0 e TLS 1.1 serão rejeitadas.

Pré-requisitos

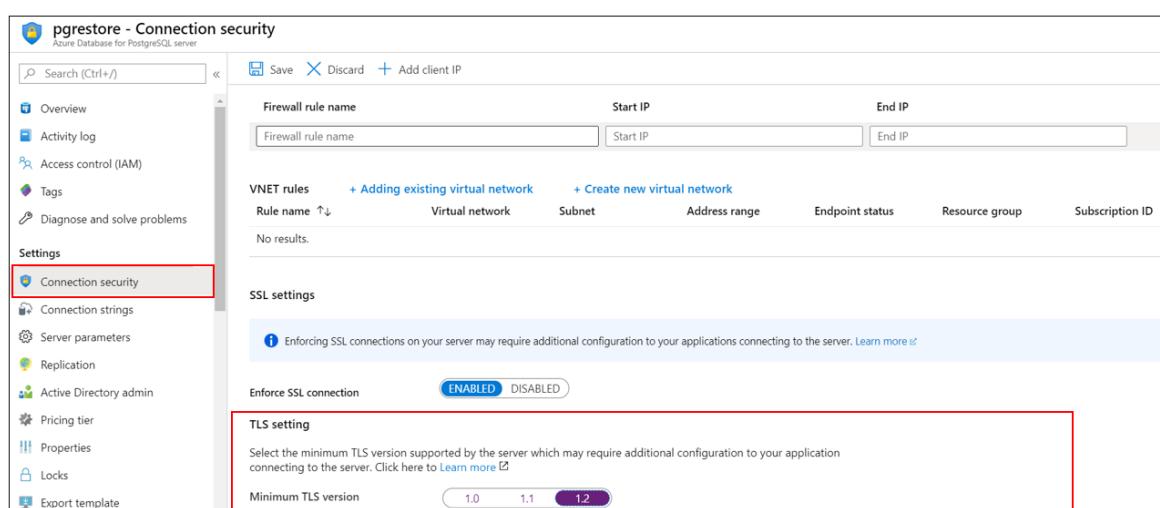
Para concluir este guia de instruções, você precisa:

- Um [Banco de Dados do Azure para PostgreSQL](#)

Definir as configurações TLS do Banco de Dados do Azure para PostgreSQL – Servidor único

Siga estas etapas para definir a versão mínima do TLS do PostgreSQL:

1. No [portal do Azure](#), selecione o Banco de Dados do Azure para PostgreSQL.
2. Na página do Banco de Dados do Azure para PostgreSQL – Servidor único, em **Configurações**, clique em **Segurança da conexão** para abrir a página de configuração de segurança da conexão.
3. Em **Versão mínima do TLS**, selecione 1.2 para negar conexões com a versão TLS inferior a TLS 1.2 para o servidor único PostgreSQL.



4. Clique em **Salvar** para salvar as alterações.

5. Uma notificação confirmará que a configuração de segurança da conexão foi habilitada com êxito.

The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'testserver'. The left sidebar lists various settings like Overview, Activity log, and Connection security (which is currently selected). The main pane displays 'Connection security' settings. Under 'SSL settings', it shows that 'Enforce SSL connection' is set to 'ENABLED'. A success message in the activity log at the top right says 'Successfully updated the connection security settings for rambotest'.

Próximas etapas

Saiba mais sobre [como criar alertas sobre métricas](#)

Como gerar uma cadeia de conexão do Banco de Dados do Azure para PostgreSQL com o PowerShell

21/05/2021 • 2 minutes to read

Este artigo demonstra como gerar uma cadeia de conexão para um servidor com Banco de Dados do Azure para PostgreSQL. Você pode usar uma cadeia de conexão para se conectar a um Banco de Dados do Azure para PostgreSQL em vários aplicativos diferentes.

Requisitos

Este artigo usa os recursos criados no seguinte guia como ponto de partida:

- [Início Rápido: Criar um servidor de Banco de Dados do Azure para PostgreSQL usando o PowerShell](#)

Obtenha a cadeia de conexão

O `Get-AzPostgreSQLConnectionString` cmdlet é usado para gerar uma cadeia de conexão a fim de conectar aplicativos ao Banco de Dados do Azure para PostgreSQL. O exemplo a seguir retorna a cadeia de conexão para um cliente PHP de `mydemoserver`.

```
Get-AzPostgreSQLConnectionString -Client PHP -Name mydemoserver -ResourceGroupName myresourcegroup
```

```
host=mydemoserver.postgres.database.azure.com port=5432 dbname={your_database} user=myadmin@mydemoserver  
password={your_password} sslmode=require
```

Os valores válidos para o parâmetro `Client` incluem:

- ADO.NET
- JDBC
- Node.js
- PHP
- Python
- Ruby
- WebApp

Próximas etapas

[Personalizar parâmetros do servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell](#)

Configurar os parâmetros do servidor no Banco de Dados do Azure para PostgreSQL – Servidor único por meio do portal do Azure

25/05/2021 • 2 minutes to read

Você pode listar, exibir e atualizar os parâmetros de configuração de um Banco de Dados do Azure para servidor PostgreSQL via portal do Azure.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- [Servidor do Banco de Dados do Azure para PostgreSQL](#)

Exibir e editar parâmetros

1. Abra o [Portal do Azure](#).
2. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL.
3. Sob a seção **configurações**, selecione **parâmetros de servidor**. A página mostra uma lista de parâmetros, valores e descrições.

The screenshot shows the Azure Portal interface for managing a PostgreSQL server named 'mydemouser'. The left sidebar has a 'Server parameters' section highlighted with a red box. The main content area displays a table of server parameters with columns for 'PARAMETER NAME', 'VALUE', and 'DESCRIPTION'. Several parameters like 'array_nulls', 'backslash_quote', and 'bytea_output' have dropdown menus open, indicating they are currently set to 'ON'. Other parameters like 'client_encoding' and 'constraint_exclusion' are set to 'SQL_ASCII' and 'PARTITION' respectively. The 'DESCRIPTION' column provides brief explanations for each setting.

PARAMETER NAME	VALUE	DESCRIPTION
array_nulls	ON OFF	Enable input of NULL elements in arrays.
backslash_quote	SAFE_ENCODING	Sets whether \" is allowed in string literals.
bytea_output	HEX	Sets the output format for bytea.
check_function_bodies	ON OFF	Check function bodies during CREATE FUNCTION.
client_encoding	SQL_ASCII	Sets the client's character set encoding.
client_min_messages	NOTICE	Sets the message levels that are sent to the client.
constraint_exclusion	PARTITION	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	0.005	Sets the planner's estimate of the cost of processing each index entry during an in
cpu_operator_cost	0.0025	Sets the planner's estimate of the cost of processing each operator or function cal
cpu_tuple_cost	0.01	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	0.1	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	iso, mdy	Sets the display format for date and time values.
deadlock_timeout	1000	Sets the amount of time, in milliseconds, to wait on a lock before checking for deadlocks.
debug_print_parse	ON OFF	Logs each query's parse tree.
debug_print_plan	ON OFF	Logs each query's execution plan.

4. Selecione o botão **suspensão** para ver os possíveis valores para parâmetros de tipo enumerado como `client_min_messages`.

PARAMETER NAME	VALUE	DESCRIPTION
array_nulls	ON	Enable input of NULL elements in arrays.
backslash_quote	SAFE_ENCODING	Sets whether "\\" is allowed in string literals.
bytea_output	HEX	Sets the output format for bytea.
check_function_bodies	ON	Check function bodies during CREATE FUNCTION.
client_encoding	SQL_ASCII	Sets the client's character set encoding.
client_min_messages	NOTICE	Sets the message levels that are sent to the client.
constraint_exclusion	DEBUG5	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	DEBUG4	Sets the planner's estimate of the cost of processing each index entry during an in
cpu_operator_cost	DEBUG3	Sets the planner's estimate of the cost of processing each operator or function cal
cpu_tuple_cost	DEBUG2	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	DEBUG1	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	LOG	Sets the display format for date and time values.
deadlock_timeout	1000	Sets the amount of time, in milliseconds, to wait on a lock before checking for dea
debug_print_parse	ON	Logs each query's parse tree.
debug_print_plan	OFF	Logs each query's execution plan.

5. Selecione ou passe o mouse sobre o botão i (informações) para ver o intervalo de valores possíveis para parâmetros numéricicos como `cpu_index_tuple_cost`.

PARAMETER NAME	VALUE	DESCRIPTION
array_nulls	ON	Enable input of NULL elements in arrays.
backslash_quote	SAFE_ENCODING	Sets whether "\\" is allowed in string literals.
bytea_output	HEX	Sets the output format for bytea.
check_function_bodies	ON	Check function bodies during CREATE FUNCTION.
client_encoding	SQL_ASCII	Sets the client's character set encoding.
client_min_messages	NOTICE	Sets the message levels that are sent to the client.
constraint_exclusion	PARTITION	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	0.005	Sets the planner's estimate of the cost of processing each index entry during an in
cpu_operator_cost	0.0025	Sets the planner's estimate of the cost of processing each operator or function cal
cpu_tuple_cost	0.01	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	0.1	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	iso, mdy	Sets the display format for date and time values.
deadlock_timeout	1000	Sets the amount of time, in milliseconds, to wait on a lock before checking for dea
debug_print_parse	ON	Logs each query's parse tree.
debug_print_plan	OFF	Logs each query's execution plan.

6. Se necessário, use a caixa de pesquisa para restringir rapidamente a um parâmetro específico. A busca está no nome e na descrição dos parâmetros.

The screenshot shows the Azure portal interface for managing a PostgreSQL server. On the left, there's a sidebar with various navigation options like Overview, Activity log, Tags, Connection security, Connection strings, Server parameters (which is selected), Pricing tier, Properties, Locks, Metrics, Alert rules, and Server logs. The main content area is titled 'mydemouser - Server parameters' and shows a table of server parameters. A red box highlights the 'max' group of parameters, which includes 'max_parallel_workers_per_gather' set to 0.

7. Altere os valores de parâmetro que você deseja ajustar. Todas as alterações feitas nessa sessão são realçadas em roxo. Depois de alterar os valores, você pode selecionar **Salvar**. Ou então, você pode **Descartar** suas alterações.

The screenshot shows the same 'Server parameters' page after changes have been made. A red box highlights the 'Save' button at the top of the page. The table now shows several parameters with their values highlighted in purple, indicating they have been modified. For example, 'cpu_index_tuple_cost' is set to 0.008.

8. Se você tiver salvo os novos valores para os parâmetros, você sempre pode reverter tudo o que fazer com os valores padrão selecionando **Redefinir tudo para o padrão**.

The screenshot shows the 'mydemouser - Server parameters' page in the Azure portal. The left sidebar includes icons for Home, Overview, Activity log, Tags, Connection security, Connection strings, Server parameters (which is selected), Pricing tier, Properties, Locks, Metrics, Alert rules, and Server logs. The main content area has a search bar, Save, Discard, and a red-highlighted 'Reset all to default' button. A table lists various server parameters with their current values and descriptions.

PARAMETER NAME	VALUE	DESCRIPTION
array_nulls	ON	Enable input of NULL elements in arrays.
backslash_quote	SAFE_ENCODING	Sets whether "\ is allowed in string literals.
bytea_output	HEX	Sets the output format for bytea.
check_function_bodies	ON	Check function bodies during CREATE FUNCTION.
client_encoding	SQL_ASCII	Sets the client's character set encoding.
client_min_messages	LOG	Sets the message levels that are sent to the client.
constraint_exclusion	PARTITION	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	0.006	Sets the planner's estimate of the cost of processing each index entry during an index scan.
cpu_operator_cost	0.0025	Sets the planner's estimate of the cost of processing each operator or function call.
cpu_tuple_cost	0.01	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	0.1	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	iso, mdy	Sets the display format for date and time values.
deadlock_timeout	1000	Sets the amount of time, in milliseconds, to wait on a lock before checking for deadlocks.
debug_print_parse	ON	Logs each query's parse tree.
debug_print_plan	ON	Logs each query's execution plan.

Próximas etapas

Saiba mais:

- [Visão geral dos parâmetros do servidor no Banco de Dados do Azure para PostgreSQL](#)
- [Configurando parâmetros usando a CLI do Azure](#)

Personalizar parâmetros de configuração de servidor do Banco de Dados do Azure para PostgreSQL – Servidor único usando a CLI do Azure

09/08/2021 • 2 minutes to read

Você pode listar, exibir e atualizar os parâmetros de configuração de um servidor PostgreSQL do Azure usando a CLI (Interface de Linha de Comando) do Azure. Um subconjunto de configurações de mecanismo é exposto no nível do servidor e pode ser modificado.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Criar um banco de dados e um servidor de Banco de Dados do Azure para PostgreSQL seguindo [Criar um Banco de Dados do Azure para o servidor PostgreSQL](#)
- Instalar a interface de linha de comando da [CLI do Azure](#) no computador ou usar o [Azure Cloud Shell](#) no portal do Azure usando seu navegador.

Listar os parâmetros de configuração de servidor para o bando de dados do Azure para servidor PostgreSQL

Para listar todos os parâmetros modificáveis em um servidor e seus valores, execute o comando [az postgres server configuration list](#).

É possível listar os parâmetros de configuração do servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`.

```
az postgres server configuration list --resource-group myresourcegroup --server mydemoserver
```

Mostrar detalhes do parâmetro de configuração do servidor

Para mostrar os detalhes sobre um parâmetro de configuração específico para um servidor, execute o comando [az postgres server configuration show](#).

Este exemplo mostra detalhes do `registro_min_mensagens` parâmetros de configuração de servidor para servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`.

```
az postgres server configuration show --name log_min_messages --resource-group myresourcegroup --server mydemoserver
```

Modificar o valor do parâmetro de configuração do servidor

Você pode modificar o valor de determinados parâmetros de configuração, que atualiza o valor da configuração subjacente para o mecanismo do servidor PostgreSQL. Para atualizar o valor de configuração, execute o comando [az postgres server configuration set](#).

Para atualizar o `registro_min_mensagens` parâmetros de configuração de servidor para servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`.

```
az postgres server configuration set --name log_min_messages --resource-group myresourcegroup --server mydemoserver --value INFO
```

Se você quiser redefinir o valor de um parâmetro de configuração, basta simplesmente optar por deixar o parâmetro opcional `--value` e o serviço aplicará o valor padrão. No exemplo acima, ele teria a seguinte aparência:

```
az postgres server configuration set --name log_min_messages --resource-group myresourcegroup --server mydemoserver
```

Esse comando redefine a configuração `log_min_messages` para o valor padrão `WARNING`. Para obter mais informações sobre a configuração do servidor e os valores permitidos, veja a documentação do PostgreSQL em [Configuração do Servidor](#).

Próximas etapas

- [Saiba como reiniciar um servidor](#)
- Para configurar e acessar os logs de servidor, confira [Logs de servidor no Banco de Dados do Azure para PostgreSQL](#)

Personalizar parâmetros do servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell

21/05/2021 • 2 minutes to read

Você pode listar, exibir e atualizar os parâmetros de configuração de um servidor do Banco de Dados do Azure para PostgreSQL usando o PowerShell. Um subconjunto de configurações de mecanismo é exposto no nível do servidor e pode ser modificado.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Do módulo [AZ do PowerShell](#) instalado localmente ou do [Azure Cloud Shell](#) no navegador
- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Se você optar por usar o PowerShell no local, conecte-se à sua conta do Azure usando o cmdlet [Connect-AzAccount](#).

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

Listar os parâmetros de configuração de servidor para o bando de dados do Azure para servidor PostgreSQL

Para listar todos os parâmetros modificáveis em um servidor e seus valores, execute o cmdlet

```
Get-AzPostgreSqlConfiguration .
```

O exemplo a seguir lista os parâmetros de configuração do servidor para o servidor **mydemoserver** no grupo de recursos **myresourcegroup**.

```
Get-AzPostgreSqlConfiguration -ResourceGroupName myresourcegroup -ServerName mydemoserver
```

Para obter a definição de cada um dos parâmetros listados, confira a seção de referência do PostgreSQL em [Variáveis do ambiente](#).

Mostrar detalhes do parâmetro de configuração do servidor

Para mostrar detalhes sobre um parâmetro de configuração específico para um servidor, execute o cmdlet

```
Get-AzPostgreSqlConfiguration
```

 e especifique o parâmetro **Nome**.

Este exemplo mostra detalhes do parâmetro de configuração de servidor **slow_query_log** para o servidor **mydemoserver** no grupo de recursos **myresourcegroup**.

```
Get-AzPostgreSqlConfiguration -Name slow_query_log -ResourceGroupName myresourcegroup -ServerName mydemoserver
```

Modificar um valor do parâmetro de configuração do servidor

Você pode modificar o valor de determinados parâmetros de configuração, que atualiza o valor da configuração subjacente para o mecanismo do servidor PostgreSQL. Para atualizar a configuração, use o cmdlet

```
Update-AzPostgreSqlConfiguration .
```

Para atualizar o parâmetro de configuração de servidor **slow_query_log** do servidor **mydemoserver** no grupo de recursos **myresourcegroup**.

```
Update-AzPostgreSqlConfiguration -Name slow_query_log -ResourceGroupName myresourcegroup -ServerName mydemoserver -Value On
```

Próximas etapas

[Aumentar automaticamente o armazenamento no Banco de Dados do Azure para PostgreSQL com o uso do PowerShell](#).

Armazenamento de aumento automático usando o portal do Azure – Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 2 minutes to read

Este artigo descreve como configurar um aumento no armazenamento do servidor do Banco de Dados do Azure para PostgreSQL sem afetar a carga de trabalho.

Quando um servidor atinge o limite de armazenamento alocado, ele é marcado como somente leitura. No entanto, se você habilitar o aumento automático do armazenamento, o armazenamento do servidor crescerá para acomodar o aumento de dados. Para servidores com menos de 100 GB de armazenamento provisionado, o tamanho do armazenamento provisionado será aumentado em 5 GB assim que o armazenamento gratuito estiver abaixo de 1 GB ou 10% do armazenamento provisionado. Para servidores com mais de 100 GB de armazenamento provisionado, o tamanho do armazenamento provisionado aumenta em 5% quando o espaço livre de armazenamento está abaixo de 10 GB do tamanho de armazenamento provisionado. [Estes](#) limites máximos de armazenamento se aplicam.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

Habilitar o aumento automático de armazenamento

Siga estas etapas para configurar o aumento automático do armazenamento do servidor PostgreSQL:

1. No [portal do Azure](#), escolha o servidor do Banco de Dados do Azure para PostgreSQL.
2. Na página do servidor PostgreSQL, em **Configurações**, clique em **Tipo de preço** para abrir a página de tipo de preço.
3. Na seção **Aumento automático**, escolha **Sim** para habilitar o aumento automático do armazenamento.

4. Clique em **OK** para salvar as alterações.

5. Uma notificação confirmará que o aumento automático foi habilitado.

Próximas etapas

Saiba mais sobre [como criar alertas sobre métricas](#).

Auto crescimento do Banco de Dados do Azure para armazenamento PostgreSQL – Servidor único usando a CLI do Azure

21/05/2021 • 2 minutes to read

Este artigo descreve como configurar um aumento no armazenamento do servidor do Banco de Dados do Azure para PostgreSQL sem afetar a carga de trabalho.

O servidor que está [alcançando o limite de armazenamento](#) está definido como somente leitura. Se o aumento automático do armazenamento estiver habilitado para servidores com menos de 100 GB de armazenamento provisionado, o tamanho do armazenamento provisionado será aumentado em 5 GB assim que o armazenamento gratuito estiver abaixo de 1 GB ou 10% do armazenamento provisionado. Para servidores com mais de 100 GB de armazenamento provisionado, o tamanho do armazenamento provisionado aumenta em 5% quando o espaço livre de armazenamento está abaixo de 10 GB do tamanho de armazenamento provisionado. [Estes](#) limites máximos de armazenamento se aplicam.

Pré-requisitos

- Você precisa de um [Banco de Dados do Azure para servidor PostgreSQL](#).
- Use o ambiente Bash no [Azure Cloud Shell](#).
[Launch Cloud Shell](#)
- Se preferir, [instale](#) a CLI do Azure para executar comandos de referência da CLI.
 - Se estiver usando uma instalação local, entre com a CLI do Azure usando o comando [az login](#). Para concluir o processo de autenticação, siga as etapas exibidas no terminal. Para mais opções de entrada, confira [Entrar com a CLI do Azure](#).
 - Quando solicitado, instale as extensões da CLI do Azure no primeiro uso. Para obter mais informações sobre extensões, confira [Usar extensões com a CLI do Azure](#).
 - Execute [az version](#) para localizar a versão e as bibliotecas dependentes que estão instaladas. Para fazer a atualização para a versão mais recente, execute [az upgrade](#).
- Este artigo exige a versão 2.0 ou posterior da CLI do Azure. Se você está usando o Azure Cloud Shell, a versão mais recente já está instalada.

Habilitar o crescimento automático do armazenamento do servidor PostgreSQL

Habilite o aumento automático do armazenamento em um servidor existente com este comando:

```
az postgres server update --name mydemoserver --resource-group myresourcegroup --auto-grow Enabled
```

Habilite o aumento automático do armazenamento enquanto cria um servidor com este comando:

```
az postgres server create --resource-group myresourcegroup --name mydemoserver --auto-grow Enabled --  
location westus --admin-user myadmin --admin-password <server_admin_password> --sku-name GP_Gen5_2 --version  
9.6
```

Próximas etapas

Saiba mais sobre [como criar alertas sobre métricas](#).

Armazenamento de crescimento automático no Banco de Dados do Azure para o servidor PostgreSQL usando o PowerShell

21/05/2021 • 2 minutes to read

Este artigo descreve como configurar um aumento no armazenamento do servidor do Banco de Dados do Azure para PostgreSQL sem afetar a carga de trabalho.

O aumento automático do armazenamento impede que o servidor [atinja o limite de armazenamento](#) e se torne somente leitura. Para servidores com 100 GB ou menos de armazenamento provisionado, o tamanho aumenta em 5 GB quando o espaço livre está abaixo de 10%. Para servidores com mais de 100 GB de armazenamento provisionado, o tamanho aumenta em 5% quando o espaço livre está abaixo de 10 GB. Os limites de armazenamento máximos se aplicam conforme especificado na seção armazenamento de [tipos de preço do Banco de Dados do Azure para PostgreSQL](#).

IMPORTANT

Não esqueça de que o armazenamento só pode ser escalado verticalmente, não horizontalmente.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Do módulo [AZ do PowerShell](#) instalado localmente ou do [Azure Cloud Shell](#) no navegador
- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Ao optar por usar o PowerShell no local, conecte-se à sua conta do Azure usando o cmdlet [Connect-AzAccount](#).

Habilitar o crescimento automático do armazenamento do servidor PostgreSQL

Habilite o aumento automático do armazenamento em um servidor existente com este comando:

```
Update-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup -StorageAutogrow Enabled
```

Habilite o aumento automático do armazenamento enquanto cria um servidor com este comando:

```
$Password = Read-Host -Prompt 'Please enter your password' -AsSecureString  
New-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup -Sku GP_Gen5_2 -StorageAutogrow  
Enabled -Location westus -AdministratorUsername myadmin -AdministratorLoginPassword $Password
```

Próximas etapas

[Como criar e gerenciar réplicas de leitura no Banco de Dados do Azure para PostgreSQL usando o PowerShell.](#)

Configurar e acessar os logs do Banco de Dados do Azure para PostgreSQL – Servidor único no portal do Azure

21/05/2021 • 2 minutes to read

É possível configurar, listar e baixar os [logs do Banco de Dados do Azure para PostgreSQL](#) no portal do Azure.

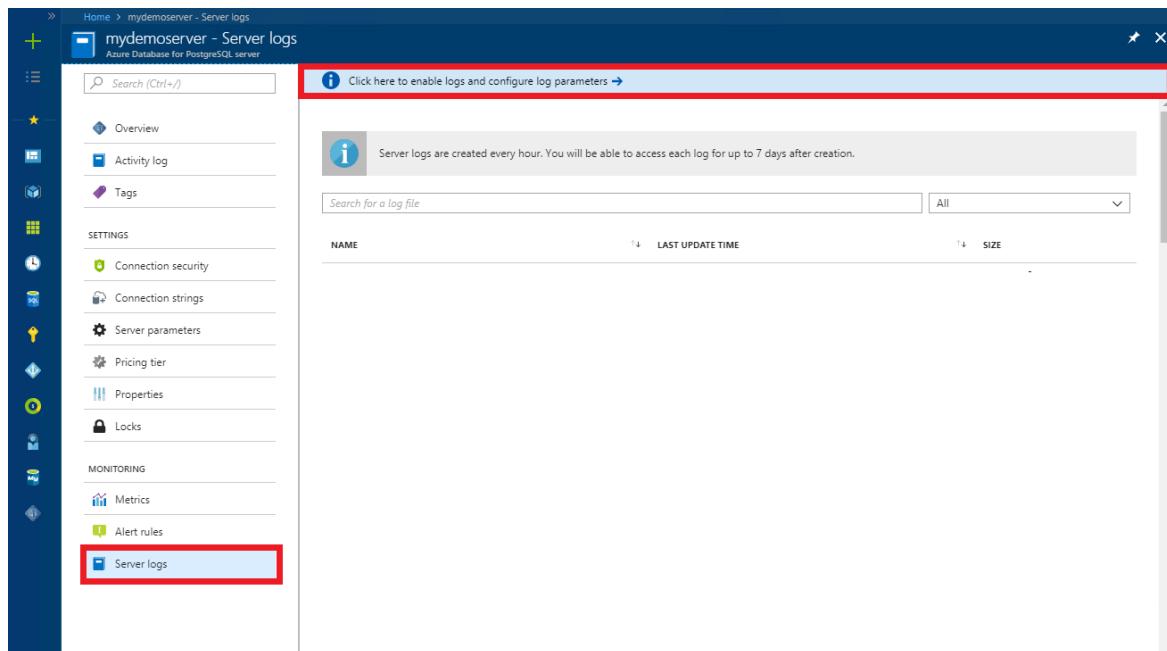
Pré-requisitos

As etapas neste artigo exigem que você tenha o [servidor do Banco de Dados do Azure para PostgreSQL](#).

Configurar o registro em log

Configure o acesso aos logs de consulta e de erros.

1. Entre no [portal do Azure](#).
2. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL.
3. Na seção **Monitoramento** da barra lateral, selecione **Logs de servidor**.



The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'mydemouser'. The left sidebar lists various monitoring and configuration options. The 'Server logs' option is highlighted with a red box. The main content area displays a message about server logs being created hourly and accessible for up to 7 days. A prominent red box highlights the 'Click here to enable logs and configure log parameters' button. Below this, there is a search bar and a table showing log files with columns for NAME, LAST UPDATE TIME, and SIZE.

4. Para ver os parâmetros do servidor, selecione **Clique aqui para habilitar logs e configurar parâmetros de log**.
5. Altere os parâmetros que você precisa ajustar. Todas as alterações feitas nessa sessão são realçadas em roxo.

Depois de alterar os parâmetros, selecione **Salvar**. Você também pode descartar as alterações.

The screenshot shows the 'Server Parameters' page in the Azure portal. At the top, there are 'Save' and 'Discard' buttons. Below them is a search bar and a table with columns: 'PARAMETER NAME', 'VALUE', and 'DESCRIPTION'. The table lists various PostgreSQL parameters like 'client_min_messages', 'debug_print_parse', etc., with their current values and descriptions.

PARAMETER NAME	VALUE	DESCRIPTION
client_min_messages	LOG	Sets the message levels that are sent to the client.
debug_print_parse	ON OFF	Logs each query's parse tree.
debug_print_plan	ON OFF	Logs each query's execution plan.
debug_print_rewritten	ON OFF	Logs each query's rewritten parse tree.
log_checkpoints	ON OFF	Logs each checkpoint.
log_connections	ON OFF	Logs each successful connection.
log_disconnections	ON OFF	Logs end of a session, including duration.
log_duration	ON OFF	Logs the duration of each completed SQL statement.
log_error_verbosity	VERBOSE	Sets the verbosity of logged messages.
log_lock_waits	ON OFF	Logs long lock waits.
log_min_duration_statement	10	Sets the minimum execution time (in milliseconds) above which statements will be logged. -1 ...
log_min_error_statement	ERROR	Causes all statements generating error at or above this level to be logged.
log_min_messages	WARNING	Sets the message levels that are logged.
log_retention_days	3	Sets how many days a log file is saved for.
log_statement	NONE	Sets the type of statements logged.

Na página **Parâmetros do Servidor**, você pode retornar à lista de logs fechando a página.

Exibir a lista e baixar os logs

Após o registro em log começar, você pode exibir uma lista dos logs disponíveis e baixar os arquivos de log individuais.

1. Abra o portal do Azure.
2. Selecione seu servidor de Banco de Dados do Azure para PostgreSQL.
3. Na seção **Monitoramento** da barra lateral, selecione **Logs de servidor**. A página exibe uma lista dos arquivos de log.

The screenshot shows the 'Server logs' page in the Azure portal. On the left, there is a sidebar with links like 'Overview', 'Activity log', 'Tags', 'SETTINGS' (with 'Connection security', 'Connection strings', 'Server parameters', 'Pricing tier', 'Properties', 'Locks'), 'MONITORING' (with 'Metrics', 'Alert rules'), and 'Server logs'. The 'Server logs' link is highlighted with a red box. The main area displays a table of logs with columns: 'NAME', 'LAST UPDATE TIME', and 'SIZE'. Each log entry has a download icon.

NAME	LAST UPDATE TIME	SIZE
postgresql-2018-02-13_030000.log	Tue, 13 Feb 2018 03:00:01 GMT	0KB
postgresql-2018-02-13_020000.log	Tue, 13 Feb 2018 02:42:50 GMT	1KB
postgresql-2018-02-13_010000.log	Tue, 13 Feb 2018 01:42:47 GMT	2KB
postgresql-2018-02-13_004150.log	Tue, 13 Feb 2018 00:42:44 GMT	3KB
postgresql-2018-02-13_004105.log	Tue, 13 Feb 2018 00:41:35 GMT	2KB
postgresql-2018-02-13_000000.log	Tue, 13 Feb 2018 00:40:44 GMT	2KB
postgresql-2018-02-12_230000.log	Mon, 12 Feb 2018 23:34:12 GMT	1KB
postgresql-2018-02-12_220000.log	Mon, 12 Feb 2018 22:34:10 GMT	1KB
postgresql-2018-02-12_210000.log	Mon, 12 Feb 2018 21:34:09 GMT	1KB
postgresql-2018-02-12_200000.log	Mon, 12 Feb 2018 20:34:07 GMT	1KB
postgresql-2018-02-12_190000.log	Mon, 12 Feb 2018 19:34:06 GMT	1KB
postgresql-2018-02-12_180000.log	Mon, 12 Feb 2018 18:33:58 GMT	1KB

TIP

A convenção de nomenclatura do log é **postgresql-yyyy-mm-dd_hh0000.log**. A data e hora usados no nome do arquivo é o momento em que o log foi emitido. Os arquivos de log são girados a cada hora ou 100 MB, o que ocorrer primeiro.

4. Se necessário, use a caixa de pesquisa para restringir rapidamente a um log específico com base na data e hora. A pesquisa busca o nome do log.

The screenshot shows the 'mydemoserver - Server logs' page in the Azure portal. On the left, there's a sidebar with various icons and a list of settings like Connection security, Connection strings, and Properties. The 'Server logs' option is selected. In the main area, there's a search bar with the placeholder 'Click here to enable logs and configure log parameters'. Below it is a message: 'Server logs are created every hour. You will be able to access each log for up to 7 days after creation.' A search bar contains the text '2-10'. To the right of the search bar is a dropdown menu set to 'All'. A red box highlights both the search bar and the list of log files below. The log file list has columns for NAME, LAST UPDATE TIME, and SIZE. Each log file name starts with 'postgresql-' followed by a date ('2018-02-10') and ends with a timestamp ('_230000.log'). The last update time is '2018-02-10T23:10:59.000Z' and the size is '1KB' for all entries.

NAME	LAST UPDATE TIME	SIZE
postgresql-2018-02-10_230000.log	"2018-02-10T23:10:59.000Z"	1KB
postgresql-2018-02-10_220000.log	"2018-02-10T22:31:31.000Z"	1KB
postgresql-2018-02-10_210000.log	"2018-02-10T21:06:43.000Z"	1KB
postgresql-2018-02-10_200000.log	"2018-02-10T20:07:37.000Z"	1KB
postgresql-2018-02-10_190000.log	"2018-02-10T19:25:52.000Z"	1KB
postgresql-2018-02-10_180000.log	"2018-02-10T18:23:59.000Z"	1KB
postgresql-2018-02-10_170000.log	"2018-02-10T17:16:36.000Z"	1KB
postgresql-2018-02-10_160000.log	"2018-02-10T16:09:26.000Z"	1KB
postgresql-2018-02-10_150000.log	"2018-02-10T15:24:18.000Z"	1KB
postgresql-2018-02-10_140000.log	"2018-02-10T14:15:21.000Z"	1KB
postgresql-2018-02-10_130000.log	"2018-02-10T13:05:17.000Z"	1KB
postgresql-2018-02-10_120000.log	"2018-02-10T12:18:27.000Z"	1KB

5. Para baixar arquivos de log individuais, selecione o ícone de seta para baixo ao lado de cada arquivo de log na linha de tabela.

The screenshot shows the same 'mydemoserver - Server logs' page as the previous one, but with a different log file list. A red box highlights the download icon (a downward arrow) next to the first log file in the list. The log file list has columns for NAME, LAST UPDATE TIME, and SIZE. The first log file is 'postgresql-2018-02-13_180000.log' with a last update time of 'Tue, 13 Feb 2018 18:42:30 GMT' and a size of '1KB'. The last update time for all other files is 'Tue, 13 Feb 2018 10:42:59 GMT'.

NAME	LAST UPDATE TIME	SIZE
postgresql-2018-02-13_180000.log	Tue, 13 Feb 2018 18:42:30 GMT	1KB
postgresql-2018-02-13_170000.log	Tue, 13 Feb 2018 17:42:28 GMT	1KB
postgresql-2018-02-13_160000.log	Tue, 13 Feb 2018 16:42:27 GMT	1KB
postgresql-2018-02-13_150000.log	Tue, 13 Feb 2018 15:42:26 GMT	1KB
postgresql-2018-02-13_140000.log	Tue, 13 Feb 2018 14:42:25 GMT	1KB
postgresql-2018-02-13_130000.log	Tue, 13 Feb 2018 13:42:23 GMT	1KB
postgresql-2018-02-13_124124.log	Tue, 13 Feb 2018 12:42:20 GMT	3KB
postgresql-2018-02-13_124035.log	Tue, 13 Feb 2018 12:41:07 GMT	2KB
postgresql-2018-02-13_120000.log	Tue, 13 Feb 2018 12:40:11 GMT	1KB
postgresql-2018-02-13_110000.log	Tue, 13 Feb 2018 11:43:01 GMT	1KB
postgresql-2018-02-13_100000.log	Tue, 13 Feb 2018 10:42:59 GMT	1KB
postgresql-2018-02-13_090000.log	Tue, 13 Feb 2018 09:42:58 GMT	1KB

Próximas etapas

- Veja [Logs do servidor de acesso na CLI](#) para saber como baixar logs programaticamente.
- Saiba mais sobre os [logs do servidor](#) no Banco de Dados do Azure para PostgreSQL.
- Para obter mais informações sobre as definições de parâmetros e o registro em log no PostgreSQL, veja a documentação do PostgreSQL em [registro em log e relatório de erros](#).

Configurar e acessar logs de servidor usando a CLI do Azure

21/05/2021 • 2 minutes to read

Você pode listar e baixar logs de erro do servidor PostgreSQL do Azure usando a interface de linha de comando (CLI do Azure). No entanto, não há suporte para acesso aos logs de transação.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- [Servidor do Banco de Dados do Azure para PostgreSQL](#)
- O utilitário de linha de comando da [CLI do Azure](#) ou o Azure Cloud Shell no navegador

Configurar o registro em log

Você pode configurar o servidor para acessar os logs de erro e os logs de consulta. Os logs de erros podem ter informações de ponto de verificação, conexão e vácuo automático.

1. Ative o registro em log.
2. Para ativar o registro em log de consulta, atualize `log_statement` e `log_min_duration_statement`.
3. Atualize o período de retenção.

Para mais informações, confira [Personalizando os parâmetros de configuração do servidor](#).

Listar logs

Para listar os arquivos de log disponíveis para o servidor, execute o comando `az postgres server-logs list`.

Você pode listar os arquivos de log para o servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`. Em seguida, direcione-os para um arquivo de texto chamado `log_files_list.txt`.

```
az postgres server-logs list --resource-group myresourcegroup --server mydemoserver > log_files_list.txt
```

Baixa logs localmente do servidor

Com o comando `az postgres server-logs download`, você pode baixar arquivos de log individuais para o seu servidor.

Use o exemplo a seguir para baixar o arquivo de log específico para o servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup` para seu ambiente local.

```
az postgres server-logs download --name 20170414-mydemoserver-postgresql.log --resource-group myresourcegroup --server mydemoserver
```

Próximas etapas

- Para saber mais sobre os logs de servidor, confira [Logs de servidor no Banco de Dados do Azure para PostgreSQL](#).
- Para saber mais sobre os parâmetros de servidor, veja [Personalizar os parâmetros de configuração de servidor usando a CLI do Azure](#).

Usar o portal do Azure para configurar alertas de métricas no Banco de Dados do Azure para PostgreSQL – Servidor único

26/05/2021 • 2 minutes to read

Este artigo mostra como configurar alertas do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure. Você pode receber um alerta com base em métricas de monitoramento para seus serviços do Azure.

O alerta é disparado quando o valor de uma métrica especificada ultrapassa um limite atribuído por você. Ele dispara tanto quando a condição é atendida pela primeira vez e quanto posteriormente, quando essa condição não está mais sendo atendida.

Você pode configurar um alerta para fazer as seguintes ações quando ele disparar:

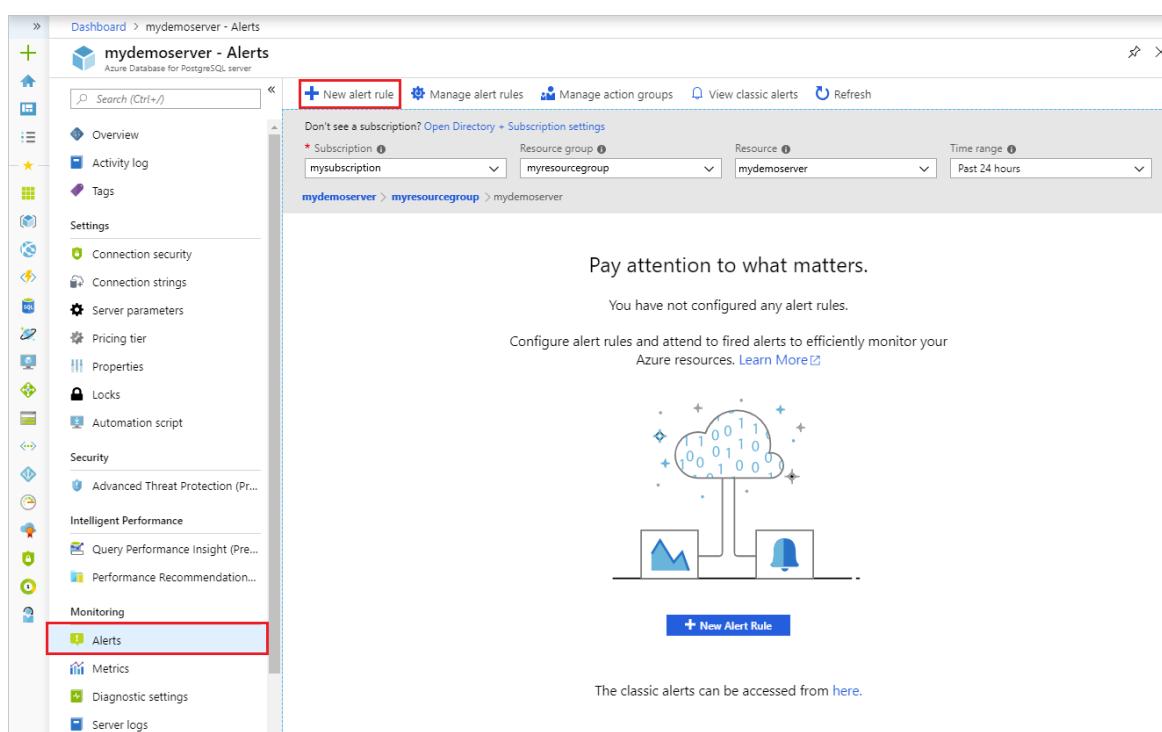
- Enviar notificações por email para o administrador e coadministradores de serviços.
- Enviar um email para outros emails que você especificar.
- Chamar um webhook.

Você pode configurar e obter informações sobre as regras de alerta usando:

- [Azure portal](#)
- [CLI do Azure](#)
- [API REST do Azure Monitor](#)

Criar uma regra de alerta em uma métrica no Portal do Azure

1. No [Portal do Azure](#), selecione o servidor do Banco de Dados do Azure para PostgreSQL que você deseja monitorar.
2. Na seção **Monitoramento** da barra lateral, selecione **Alertas** como mostrado abaixo:



3. Selecione Adicionar alerta de métrica (ícone +).

4. A página Criar regra é aberta, conforme mostrado abaixo. Preencha as informações obrigatórias:

The screenshot shows the 'Create rule' interface. On the left, there's a sidebar with various icons. The main area has sections for 'RESOURCE' (set to 'mydemouser'), 'HIERARCHY' (showing 'mysubscription > myresourcegroup'), 'CONDITION' (with a note: 'No condition defined, click on 'Add condition' to select a signal and define its logic'), 'ACTION GROUPS' (with a note: 'Notify your team via email and text messages or automate actions using webhooks, runbooks, functions, logic apps or integrating with external ITSM solutions. Learn more [here](#)'), and 'ALERT DETAILS' (with fields for 'Alert rule name' and 'Description'). A red box highlights the 'Add condition' button under the 'CONDITION' section.

5. Dentro da seção **Condição**, selecione **Adicionar condição**.

6. Selecione uma métrica da lista de sinais sobre a qual deseja ser alertado. Neste exemplo, selecione "Porcentagem de armazenamento".

The screenshot shows the 'Create rule' interface with the 'CONDITION' section selected. A modal window titled 'Configure signal logic' is open, displaying a list of signals. The 'All signals (46)' section is visible at the top. A red box highlights the 'Storage percent' row in the list, which is described as a Metric signal type for the Platform monitor service. Other signals listed include CPU percent, Memory percent, IO percent, Storage used, Storage limit, Server Log storage percent, Server Log storage used, Server Log storage limit, Active Connections, Failed Connections, Backup Storage used, Network Out, Network In, and Replica Lag.

7. Configure a lógica de alerta, incluindo a **Condição** (por exemplo, "Maior que"), o **Limite** (por exemplo, 85%), a **Agregação de Tempo**, o **Período** durante o qual a regra de métrica deverá ser atendida antes de o alerta disparar (por exemplo, "Os últimos 30 minutos") e **Frequência**.

Selecione **Concluído** ao concluir.

The screenshot shows the 'Create rule' interface in the Azure portal. On the left, there's a sidebar with various icons for different services like Storage, Compute, and Monitoring. The main area is titled 'Create rule' under 'Rules management'. It has sections for 'RESOURCE' (selected 'mydemouser'), 'HIERARCHY' (mysubscription > myresourcegroup), 'CONDITION' (no condition defined, with a link to 'Add condition'), 'ACTION GROUPS' (no action group selected, with 'Select existing' and 'Create New' buttons), and 'ALERT DETAILS' (Alert rule name: 'Specify alert rule name. Sample: 'Percentage CPU greater than 70'', Description: 'Specify alert description here...'). On the right, a 'Configure signal logic' panel is open, showing a chart titled 'Storage percent(Platform)' with a value of 10.43% over the last 6 hours. Below the chart is the 'Alert logic' configuration, which includes a red box highlighting the 'Condition' section: 'Greater than' (Time Aggregation: Average, Threshold: 85%), 'Evaluated based on' (Period grain: Over the last 30 minutes, Frequency: Every 1 Minute), and a 'Done' button.

8. Dentro da seção **Grupos de Ações**, selecione **Criar Novo** para criar um novo grupo para receber notificações sobre o alerta.
9. Preencha o formulário "Adicionar grupo de ações" com um nome, o nome curto, a assinatura e o grupo de recursos.
10. Configure o tipo de ação **Email/SMS/Push/Voz**.

Escolha "Enviar email para a Função do Azure Resource Manager" para selecionar os Proprietários da assinatura, Colaboradores e Leitores para receber notificações.

Opcionalmente, forneça um URI válido no campo **Webhook** se você quiser chamá-lo quando o alerta for disparado.

Selecione **OK** ao concluir.

The screenshot shows two overlapping windows. The top window is titled 'Add action group' and contains fields for Action group name (newactiongroup), Short name (group1), Subscription (mysubscription), and Resource group (myresourcegroup). Below these are sections for Actions, Privacy Statement, and Pricing. The bottom window is titled 'Email/SMS/Push/Voice' and shows configuration for Email, SMS, Azure app Push Notifications, and Voice. It includes fields for Name (email), Email (email@contoso.com), Country code (1), Phone number (1234567890), and a note about carrier charges. An 'OK' button is at the bottom.

11. Especifique um Nome da regra de alerta, uma Descrição e uma Gravidade.

The screenshot shows the 'Create rule' dialog under 'Rules management'. It displays 'ACTION GROUPS' with 'newactiongroup' selected. The 'ALERT DETAILS' section is highlighted with a red box and contains fields for Alert rule name (Storage percentage greater than 85), Description (Storage percentage greater than 85), Severity (Sev 3), and a toggle for Enable rule upon creation (Yes). A note below states: 'It can take up to 10 minutes for a metric alert rule to become active.' A 'Create alert rule' button is at the bottom.

12. Selecione Criar regra de alerta para criar o alerta.

Em alguns minutos, o alerta estará ativo e disparará conforme descrito anteriormente.

Gerenciar seus alertas

Depois de criar um alerta, você poderá selecioná-lo e executar as seguintes ações:

- Exibir um gráfico mostrando o limite de métrica e os valores reais do dia anterior relevante para este alerta.
- **Editar** ou **Excluir** a regra de alerta.
- **Desabilitar** ou **Habilitar** o alerta, se desejar interromper temporariamente ou retomar o recebimento de notificações.

Próximas etapas

- Saiba mais sobre como [configurar webhooks em alertas](#).
- Tenha uma [visão geral da coleção de métricas](#) para verificar se o serviço está disponível e responsivo.

Solucionar problemas de conexões no Banco de Dados do Azure para PostgreSQL – Servidor único

21/05/2021 • 3 minutes to read

Os problemas de conexão podem ser causados por diversas coisas, incluindo:

- Configurações de firewall
- Tempo limite da conexão
- Informações de logon incorretas
- Atingido o limite máximo em alguns recursos do Banco de Dados do Azure para PostgreSQL
- Problemas com a infraestrutura do serviço
- Manutenção executada no serviço
- A alocação de computação do servidor é alterada pelo dimensionamento do número de vCores ou pela movimentação para outra camada de serviço

Normalmente, problemas de conexão com o Banco de Dados do Azure para PostgreSQL podem ser classificados da seguinte forma:

- Erros transitórios (de curta duração ou intermitentes)
- Erros persistentes ou não transitórios (erros regularmente recorrentes)

Solucionar problemas de erros transitórios

Quando a manutenção é executada, o sistema encontra um erro com o hardware ou software ou se você altera a camada de serviço ou vCores do seu servidor, ocorrem erros transitórios. O serviço Banco de Dados do Azure para PostgreSQL tem alta disponibilidade interna e foi projetado para atenuar esses tipos de problemas automaticamente. No entanto, seu aplicativo perde sua conexão ao servidor por um curto período de tempo de geralmente menos de 60 segundos, no máximo. Alguns eventos ocasionalmente podem levar mais tempo para serem corrigidos, como quando uma transação grande causa uma recuperação de execução longa.

Etapas para resolver problemas de conectividade temporários

1. Confira o [Painel de Serviços do Microsoft Azure](#) quanto a quaisquer interrupções conhecidas que tenham ocorrido durante o tempo em que o erro foi relatado pelo aplicativo.
2. Os aplicativos que se conectam a um serviço de nuvem, como Banco de Dados do Azure para PostgreSQL, devem esperar eventos transitórios de reconfiguração periódicos e implementar lógica de repetição para lidar com esses erros, em vez de exibir esses erros como erros de aplicativo aos usuários. Examine [Tratamento de erros de conectividade transitória para o Banco de Dados do Azure para PostgreSQL](#) para práticas recomendadas e diretrizes de design para tratar erros transitórios.
3. Conforme um servidor se aproxima dos limites de recursos, os erros podem parecer um problema de conectividade transitório. Consulte [Limitações no Banco de Dados do Azure para PostgreSQL](#).
4. Se problemas de conectividade continuarem, se a duração pela qual o aplicativo encontra o erro exceder 60 segundos ou se você vir várias ocorrências do erro em um determinado dia, envie uma solicitação de suporte do Azure selecionando **Obter Suporte** no site de [Suporte do Azure](#).

Solucionar erros persistentes

Se o aplicativo falhar persistentemente em se conectar ao Banco de Dados do Azure para PostgreSQL, ele normalmente indicará um problema com um dos seguintes:

- Configuração do firewall do servidor: configure o firewall do Banco de Dados do Azure para PostgreSQL para permitir conexões de seu cliente, incluindo servidores proxy e gateways.
- Configuração do firewall do cliente: o firewall do cliente deve permitir conexões com o servidor de banco de dados. É necessário permitir endereços IP e portas do servidor nas quais você não pode se conectar, além dos nomes de aplicativo, como PostgreSQL, em alguns firewalls.
- Erro do usuário: talvez você tenha digitado incorretamente parâmetros de conexão, como o nome do servidor na cadeia de conexão, ou tenha esquecido o sufixo `@servername` no nome de usuário.
- Se você vir o erro *O servidor não está configurado para permitir conexões IPv6*, observe que a camada Básico não permite pontos de extremidade de serviço de VNet. É preciso remover o ponto de extremidade Microsoft.Sql da sub-rede que está tentando se conectar ao servidor Básico.
- Quando aparece o erro de conexão *Valor de sslmode "****" inválido quando o suporte a SSL não está integrado*, isso significa que o cliente PostgreSQL não oferece suporte para SSL. Provavelmente, a libpq do lado do cliente não foi compilada com o sinalizador `--with-openssl`. Tente se conectar a um cliente PostgreSQL que dê suporte para SSL.

Etapas para resolver os problemas de conectividade temporários

1. Configure as [regras de firewall](#) para permitir o endereço IP do cliente. Para fins de testes temporários, configure uma regra de firewall usando 0.0.0.0 como o endereço IP inicial e usando 255.255.255.255 como o endereço IP final. Isso abrirá o servidor para todos os endereços IP. Se isso resolver seu problema de conectividade, remova essa regra e crie uma regra de firewall para um intervalo de endereçamento ou um endereço IP adequadamente limitado.
2. Em todos os firewalls entre o cliente e a Internet, abra a porta 5432 para conexões de saída.
3. Verifique a cadeia de conexão e outras configurações de conexão.
4. Verifique a integridade do serviço no painel. Se você achar que há uma interrupção regional, consulte [Visão geral da continuidade dos negócios com o Banco de Dados do Azure para PostgreSQL](#) para obter as etapas para recuperar para uma nova região.

Próximas etapas

- [Manipulação de erros de conectividade transitória para Banco de Dados do Azure para PostgreSQL](#)

Solucionar problemas de criptografia de dados no banco de dado do Azure para PostgreSQL – servidor único

21/05/2021 • 2 minutes to read

Este artigo ajuda a identificar e resolver problemas comuns que podem ocorrer na implantação de servidor único do Banco de Dados do Azure para PostgreSQL quando configurado com criptografia de dados usando uma chave gerenciada pelo cliente.

Introdução

Quando você configura a criptografia de dados para usar uma chave gerenciada pelo cliente no Azure Key Vault, o servidor requer acesso contínuo à chave. Se o servidor perder o acesso à chave gerenciada pelo cliente no Azure Key Vault, ele negará todas as conexões, retornará a mensagem de erro apropriada e alterará seu estado para *Inacessível* no portal do Azure.

Se você não precisar mais de um servidor de Banco de Dados do Azure para PostgreSQL inacessível, poderá excluí-lo para evitar custos. Nenhuma outra ação no servidor é permitida até que o acesso ao cofre das chaves seja restaurado e o servidor esteja disponível. Também não é possível alterar a opção de criptografia de dados de (gerenciada pelo cliente) para (gerenciada pelo serviço) em um servidor inacessível quando ele é criptografado com uma chave gerenciada pelo cliente. Você terá que revalidar a chave manualmente antes que o servidor esteja acessível mais uma vez. Esta ação é necessária para proteger os dados contra o acesso não autorizado enquanto as permissões para a chave gerenciada pelo cliente forem revogadas.

Erros comuns que fazem com que o servidor se torne inacessível

As configurações incorretas a seguir causam a maioria dos problemas com criptografia de dados que usam chaves do Azure Key Vault:

- O cofre de chaves não está disponível ou não existe:
 - O cofre de chaves foi excluído por engano.
 - Um erro de rede intermitente faz com que o cofre de chaves fique indisponível.
- Você não tem permissão para acessar o cofre de chaves ou a chave não existe:
 - A chave expirou ou foi excluída ou desabilitada acidentalmente.
- A identidade gerenciada da instância do Banco de Dados do Azure para PostgreSQL foi excluída acidentalmente.
 - A identidade gerenciada da instância do Banco de Dados do Azure para PostgreSQL não tem permissões de chave suficientes. Por exemplo, as permissões não incluem Obter, Encapsular e Cancelar encapsulamento.
 - As permissões de identidade gerenciadas para a instância do Banco de Dados do Azure para PostgreSQL foram revogadas ou excluídas.

Identificar e resolver erros comuns

Erros no cofre de chaves

Cofre de chaves desabilitado

- `AzureKeyVaultKeyDisabledMessage`
- **Explicação:** a operação não pôde ser concluída no servidor porque a chave do Azure Key Vault está desabilitada.

Permissões de cofre de chaves ausentes

- `AzureKeyVaultMissingPermissionsMessage`
- **Explicação:** o servidor não tem as permissões Obter, Encapsular e Cancelar encapsulamento necessárias para o Azure Key Vault. Conceda quaisquer permissões ausentes à entidade de serviço com ID.

Atenuação

- Confirme se a chave gerenciada pelo cliente está presente no cofre de chaves.
- Identifique o cofre de chaves e vá até ele no portal do Azure.
- Verifique se o URI da chave identifica uma chave que está presente.

Próximas etapas

[Usar o portal do Azure para configurar a criptografia de dados com uma chave gerenciada pelo cliente no Banco de Dados do Azure para PostgreSQL](#)

Criar e gerenciar regras e pontos de extremidade de serviço de VNet no Banco de Dados do Azure para PostgreSQL – Servidor Único usando o portal do Azure

21/05/2021 • 4 minutes to read

As regras e pontos de extremidade de serviços de VNet (rede virtual) estendem o espaço de endereço privado de uma rede virtual para seu servidor do Banco de Dados do Azure para PostgreSQL. Para obter uma visão geral dos pontos de extremidade de serviço de VNet do Banco de Dados do Azure para PostgreSQL, confira [Pontos de extremidade de serviço de VNet do servidor do Banco de Dados do Azure para PostgreSQL](#). Os terminais de serviços da VNet estão disponíveis em todas as regiões suportadas para o Banco de Dados do Azure para PostgreSQL.

NOTE

O suporte para ponto de extremidade de serviço de VNet é apenas para servidores de Uso Geral e Otimizados para Memória. No caso de emparelhamento de VNet, se o tráfego estiver fluindo através de um Gateway VNet comum com pontos de extremidade de serviço e deve fluir para o par, crie uma regra ACL/VNet para permitir que Máquinas Virtuais do Azure accessem o Banco de Dados do Azure para servidor PostgreSQL.

Criar uma regra de VNet e habilitar pontos de extremidade de serviço no portal do Azure

1. Na página do servidor PostgreSQL, no título Configurações, clique em **Segurança de Conexão** para abrir o painel Segurança de Conexão para o Banco de Dados do Azure para PostgreSQL.
2. Verifique se o controle Permitir acesso aos serviços do Azure está definido como **DESATIVADO**.

IMPORTANT

Se você deixar o controle definido como **ATIVADO**, o servidor do Banco de Dados do Azure para PostgreSQL aceitará comunicação de qualquer sub-rede. Deixar o controle definido como **ON** pode ocasionar acesso excessivo de um ponto de vista de segurança. O recurso de ponto de extremidade do serviço de rede virtual do Microsoft Azure, em conjunto com o recurso de regra da rede virtual do Banco de Dados do Azure para PostgreSQL, pode reduzir a área da superfície de segurança.

3. Em seguida, clique em **+** **Adicionar rede virtual existente**. Se você não tiver uma VNet existente, clique em **+** **Criar nova rede virtual** para criar uma. Confira [Início Rápido: criar uma rede virtual usando o portal do Azure](#)

4. Insira um nome de regra de VNet, selecione a assinatura, a rede virtual e o nome da sub-rede. Depois, clique em **Habilitar**. Dessa forma, os pontos de extremidade de serviço da VNet serão habilitados na sub-rede usando a marcação de serviço **Microsoft.SQL**.

A conta deve ter as permissões necessárias para criar uma rede virtual e um ponto de extremidade de serviço.

Pontos de extremidade de serviço podem ser configurados em redes virtuais de forma independente por um usuário com acesso de gravação à rede virtual.

Para proteger os recursos de serviço do Azure em uma VNet, o usuário deve ter permissão para "Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/" das sub-redes que estão sendo adicionadas. Essa permissão está incluída nas funções de administrador de serviço internas por padrão e pode ser modificada com a criação de funções personalizadas.

Saiba mais sobre [funções internas](#) e como atribuir permissões específicas a [funções personalizadas](#).

As VNets e os recursos de serviço do Azure podem estar na mesma assinatura ou em assinaturas

diferentes. Se os recursos de serviço VNet e Azure estiverem em assinaturas diferentes, os recursos deverão estar no mesmo locatário do Active Directory (AD). As duas assinaturas devem ter o provedor de recursos **Microsoft.Sql** registrado. Para obter mais informações, confira [resource-manager-registration](#)

IMPORTANT

É altamente recomendável ler este artigo sobre considerações e configurações de ponto de extremidade de serviço antes de configurá-los. **Ponto de extremidade de serviço de Rede Virtual: Um ponto de extremidade de serviço de Rede Virtual** é uma sub-rede cujos valores de propriedade incluem um ou mais nomes formais de tipo de serviço do Azure. Os pontos de extremidade de serviços de VNet usam o nome de tipo de serviço **Microsoft.Sql**, que se refere ao serviço do Azure chamado Banco de Dados SQL. Essa marcação de serviço também é aplicável aos serviços Banco de Dados SQL do Azure, Banco de Dados do Azure para PostgreSQL e MySQL. É importante observar que, ao aplicar a marcação de serviço **Microsoft.Sql** a um ponto de extremidade de serviço de VNet, ela configura o tráfego do ponto de extremidade de serviço para todos os serviços de Banco de Dados do Azure, incluindo servidores do Banco de Dados SQL do Azure, do Banco de Dados do Azure para PostgreSQL e do Banco de Dados do Azure para MySQL na sub-rede.

- Depois de habilitá-lo, clique em OK. Você verá que os pontos de extremidade de serviço da VNet estão habilitados junto com uma regra de VNet.

The screenshot shows the Azure portal interface for managing connection security. On the left, there's a sidebar with navigation links like Home, Overview, Activity log, Tags, Connection strings, Server parameters, Pricing tier, Properties, Locks, Metrics, Alerts (classic), and Server logs. The main area is titled 'mydemoserver - Connection security' and shows the 'Connection security' tab selected. It has sections for Firewall rules, VNET Rules, and SSL settings. The VNET Rules section is highlighted with a red box and contains a single row:

RULE NAME	VIRTUAL NETWORK	SUBNET	ADDRESS RANGE	ENDPOINT STATUS	RESOURCE GROUP	SUBSCRIPTION ID	STATE
vnet-test	vnet-test-vnet	default	172.16.136.0/24	Enabled	vnet-test	ffffffff-ffff-ffff-ffff-...	Ready

Próximas etapas

- Da mesma forma, é possível criar scripts para [Habilitar pontos de extremidade de serviço VNet e criar um regra VNET para o Banco de Dados do Azure para PostgreSQL usando a CLI do Azure](#).
- Para obter ajuda com a conexão com um Banco de Dados para servidor PostgreSQL, veja [Bibliotecas de conexão para o Banco de Dados do Azure para PostgreSQL](#)

Criar e gerenciar pontos de extremidade de serviço da VNet para o Banco de Dados do Azure para PostgreSQL – Servidor único usando a CLI do Azure

21/05/2021 • 6 minutes to read

As regras e pontos de extremidade de serviços de VNet (rede virtual) estendem o espaço de endereço privado de uma rede virtual para seu servidor do Banco de Dados do Azure para PostgreSQL. Usando comandos convenientes da CLI (interface de linha de comando) do Azure, você pode criar, atualizar, excluir, listar e mostrar as regras e os pontos de extremidade de serviço de VNet para gerenciar o servidor. Para obter uma visão geral dos pontos de extremidade de serviço de VNet do Banco de Dados do Azure para PostgreSQL, confira [Pontos de extremidade de serviço de VNet do servidor do Banco de Dados do Azure para PostgreSQL](#). Os terminais de serviços da VNet estão disponíveis em todas as regiões suportadas para o Banco de Dados do Azure para PostgreSQL.

Se você não tiver uma [assinatura do Azure](#), crie uma [conta gratuita](#) antes de começar.

Pré-requisitos

Para seguir este guia de instruções:

- Instalar [CLI do Azure](#) ou use o Azure Cloud Shell no navegador.
- Crie um [Banco de dados e servidor do Banco de Dados do Azure para PostgreSQL](#).

NOTE

O suporte para ponto de extremidade de serviço de VNet é apenas para servidores de Uso Geral e Otimizados para Memória. No caso de emparelhamento de VNet, se o tráfego estiver fluindo através de um Gateway VNet comum com pontos de extremidade de serviço e deve fluir para o par, crie uma regra ACL/VNet para permitir que Máquinas Virtuais do Azure acessem o Banco de Dados do Azure para servidor PostgreSQL.

- Use o ambiente Bash no [Azure Cloud Shell](#).

- Se preferir, [instale](#) a CLI do Azure para executar comandos de referência da CLI.
 - Se estiver usando uma instalação local, entre com a CLI do Azure usando o comando [az login](#). Para concluir o processo de autenticação, siga as etapas exibidas no terminal. Para mais opções de entrada, confira [Entrar com a CLI do Azure](#).
 - Quando solicitado, instale as extensões da CLI do Azure no primeiro uso. Para obter mais informações sobre extensões, confira [Usar extensões com a CLI do Azure](#).
 - Execute [az version](#) para localizar a versão e as bibliotecas dependentes que estão instaladas. Para fazer a atualização para a versão mais recente, execute [az upgrade](#).
- Este artigo exige a versão 2.0 ou posterior da CLI do Azure. Se você está usando o Azure Cloud Shell, a versão mais recente já está instalada.

Configurar pontos de extremidade de serviço de VNet para Banco de Dados do Azure para PostgreSQL

Os comandos `az network vnet` são usados para configurar redes virtuais.

Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Selecione a ID da assinatura específica em sua conta usando o comando `az account set`. Substitua a propriedade `id` da saída `logon az` para a sua assinatura no espaço reservado da ID da assinatura.

- A conta deve ter as permissões necessárias para criar uma rede virtual e um ponto de extremidade de serviço.

Pontos de extremidade de serviço podem ser configurados em redes virtuais de forma independente por um usuário com acesso de gravação à rede virtual.

Para proteger os recursos de serviço do Azure em uma VNet, o usuário deve ter permissão para `"Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/"` das sub-redes que estão sendo adicionadas. Essa permissão está incluída nas funções de administrador de serviço internas por padrão e pode ser modificada com a criação de funções personalizadas.

Saiba mais sobre [funções internas](#) e como atribuir permissões específicas a [funções personalizadas](#).

As VNets e os recursos de serviço do Azure podem estar na mesma assinatura ou em assinaturas diferentes. Se os recursos de serviço VNet e Azure estiverem em assinaturas diferentes, os recursos deverão estar no mesmo locatário do Active Directory (AD). As duas assinaturas devem ter o provedor de recursos **Microsoft.Sql** registrado. Para saber mais, confira [resource-manager-registration](#).

IMPORTANT

É altamente recomendável ler este artigo sobre considerações e configurações de ponto de extremidade de serviço antes de executar o script de exemplo abaixo ou configurar pontos de extremidade de serviço. **Ponto de extremidade de serviço de Rede Virtual:** Um [ponto de extremidade de serviço de Rede Virtual](#) é uma sub-rede cujos valores de propriedade incluem um ou mais nomes formais de tipo de serviço do Azure. Os pontos de extremidade de serviços de VNet usam o nome de tipo de serviço **Microsoft.Sql**, que se refere ao serviço do Azure chamado Banco de Dados SQL. Essa marcação de serviço também é aplicável aos serviços Banco de Dados SQL do Azure, Banco de Dados do Azure para PostgreSQL e MySQL. É importante observar que, ao aplicar a marcação de serviço **Microsoft.Sql** a um ponto de extremidade de serviço de VNet, ela configura o tráfego do ponto de extremidade de serviço para todos os serviços de Banco de Dados do Azure, incluindo servidores do Banco de Dados SQL do Azure, do Banco de Dados do Azure para PostgreSQL e do Banco de Dados do Azure para MySQL na sub-rede.

Exemplo de script para criar um Banco de Dados do Azure para PostgreSQL, uma VNet, um ponto de extremidade de serviço de VNet e proteger o servidor da sub-rede com uma regra de VNet

Neste script de exemplo, altere as linhas destacadas para personalizar o nome de usuário administrador e a senha. Substitua a `SubscriptionID` usada nos comandos `az account set --subscription` pelo identificador de assinatura.

```

#!/bin/bash

# To find the name of an Azure region in the CLI run this command: az account list-locations
# Substitute <subscription id> with your identifier
az account set --subscription <subscription id>

# Create a resource group
az group create \
--name myresourcegroup \
--location westus

# Create a PostgreSQL server in the resource group
# Name of a server maps to DNS name and is thus required to be globally unique in Azure.
# Substitute the <server_admin_password> with your own value.
az postgres server create \
--name mydemoserver \
--resource-group myresourcegroup \
--location westus \
--admin-user mylogin \
--admin-password <server_admin_password> \
--sku-name GP_Gen4_2

# Get available service endpoints for Azure region output is JSON
# Use the command below to get the list of services supported for endpoints, for an Azure region, say
#"westus".
az network vnet list-endpoint-services \
-l westus

# Add Azure SQL service endpoint to a subnet *mySubnet* while creating the virtual network *myVNet* output
# is JSON
az network vnet create \
-g myresourcegroup \
-n myVNet \
--address-prefixes 10.0.0.0/16 \
-l westus

# Creates the service endpoint
az network vnet subnet create \
-g myresourcegroup \
-n mySubnet \
--vnet-name myVNet \
--address-prefix 10.0.1.0/24 \
--service-endpoints Microsoft.SQL

# View service endpoints configured on a subnet
az network vnet subnet show \
-g myresourcegroup \
-n mySubnet \
--vnet-name myVNet

# Create a VNet rule on the sever to secure it to the subnet. Note: resource group (-g) parameter is where
# the database exists. VNet resource group if different should be specified using subnet id (URI) instead of
# subnet, VNet pair.
az postgres server vnet-rule create \
-n myRule \
-g myresourcegroup \
-s mydemoserver \
--vnet-name myVNet \
--subnet mySubnet

```

Limpar a implantação

Após a execução do script de exemplo, o comando a seguir pode ser usado para remover o grupo de recursos e todos os recursos associados a ele.

```
#!/bin/bash
az group delete --name myresourcegroup
```

Criar e gerenciar o Link Privado para o Banco de Dados do Azure para PostgreSQL – Servidor Único usando o portal

01/07/2021 • 7 minutes to read

Um ponto de extremidade privado é o bloco de construção fundamental para o link privado no Azure. Ele permite que os recursos do Azure, como VMs (máquinas virtuais), se comuniquem de forma privada com recursos de link privado. Neste artigo, você aprenderá a usar o portal do Azure para criar uma VM em uma rede virtual do Azure e um Servidor Único do Banco de Dados do Azure para PostgreSQL com um ponto de extremidade privado do Azure.

Se você não tiver uma assinatura do Azure, crie uma [conta gratuita](#) antes de começar.

NOTE

O recurso de link privado está disponível apenas para servidores do Bancos de Dados do Azure para PostgreSQL nas camadas de preços de Uso geral ou Otimizado para Memória. Assegure-se de que o servidor de banco de dados esteja em um desses níveis de preços.

Entrar no Azure

Entre no [portal do Azure](#).

Criar uma VM do Azure

Nesta seção, você criará uma rede virtual e a sub-rede para hospedar a VM usada para acessar seu recurso de Link Privado (um PostgreSQL no Azure).

Criar a rede virtual

Nesta seção, você criará uma rede virtual e a sub-rede para hospedar a VM usada para acessar o recurso de Link Privado.

1. No canto superior esquerdo da tela, selecione **Criar um recurso > Rede > Rede virtual**.
2. Em **Criar rede virtual**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
Nome	Insira <i>MyVirtualNetwork</i> .
Espaço de endereço	Insira <i>10.1.0.0/16</i> .
Subscription	Seleciona sua assinatura.
Resource group	Seleciona Criar novo e insira <i>myResourceGroup</i> , depois selecione OK.
Local	Seleciona Europa Ocidental .

CONFIGURAÇÃO	VALOR
Sub-rede – Nome	Insira <i>mySubnet</i> .
Sub-rede – Intervalo de endereços	Insira <i>10.1.0.0/24</i> .

3. Deixe o restante com os valores padrão e selecione **Criar**.

Criar máquina virtual

1. No lado superior esquerdo da tela no portal do Azure, selecione **Criar um recurso > Computação > Máquina Virtual**.
2. Em **Criar uma máquina virtual – Noções básicas**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
DETALHES DO PROJETO	
Subscription	Selecione sua assinatura.
Resource group	Selecione myResourceGroup . Você o criou na seção anterior.
DETALHES DA INSTÂNCIA	
Nome da máquina virtual	Insira <i>myVm</i> .
Região	Selecione Europa Ocidental .
Opções de disponibilidade	Deixe o padrão Nenhuma redundância de infraestrutura necessária .
Imagen	Selecione Windows Server 2019 Datacenter .
Tamanho	Deixe o padrão Standard DS1 v2 .
CONTA DE ADMINISTRADOR	
Nome de Usuário	Insira um nome de usuário de sua escolha.
Senha	Insira uma senha de sua escolha. A senha deve ter no mínimo 12 caracteres e atender a requisitos de complexidade definidos .
Confirmar Senha	Reinsira a senha.
REGRAS DE PORTA DE ENTRADA	
Porta de entrada públicas	Deixar o padrão Nenhum .
ECONOMIZE DINHEIRO	

CONFIGURAÇÃO	VALOR
Já tem uma licença do Windows?	Deixe o padrão Não .

3. Selecione **Avançar: Discos**.

4. Em **Criar uma máquina virtual – Discos**, mantenha os padrões e selecione **Avançar: Rede**.

5. Em **Criar uma máquina virtual – Rede**, selecione estas informações:

CONFIGURAÇÃO	VALOR
Rede virtual	Deixe o padrão MyVirtualNetwork .
Espaço de endereço	Deixar o padrão 10.1.0.0/24 .
Sub-rede	Deixar o padrão mySubnet (10.1.0.0/24) .
IP público	Deixe o padrão (novo) myVm-ip .
Porta de entrada públicas	Selecionar Permitir portas selecionadas .
Selecionar as portas de entrada	Selecionar HTTP e RDP .

6. Selecione **Examinar + criar**. Você é levado até a página **Examinar + criar**, na qual o Azure valida sua configuração.

7. Quando vir a mensagem **Validação aprovada**, selecione **Criar**.

NOTE

Em alguns casos, o Banco de Dados do Azure para PostgreSQL e a sub-rede da VNet estão em assinaturas diferentes. Nesses casos, você deve garantir as seguintes configurações:

- Verifique se as duas assinaturas têm o provedor de recursos **Microsoft.DBforPostgreSQL** registrado. Para obter mais informações, confira [resource-manager-registration](#)

Criar um Servidor Único do Banco de Dados do Azure para PostgreSQL

Nesta seção, você criará um servidor de Banco de Dados do Azure para PostgreSQL no Azure.

1. No lado superior esquerdo da tela no portal do Azure, selecione **Criar um recurso > Bancos de Dados > Banco de Dados do Azure para PostgreSQL**.
2. Na opção de implantação **Banco de Dados do Azure para PostgreSQL**, selecione **Servidor único** e forneça estas informações:

CONFIGURAÇÃO	VALOR
Detalhes do projeto	

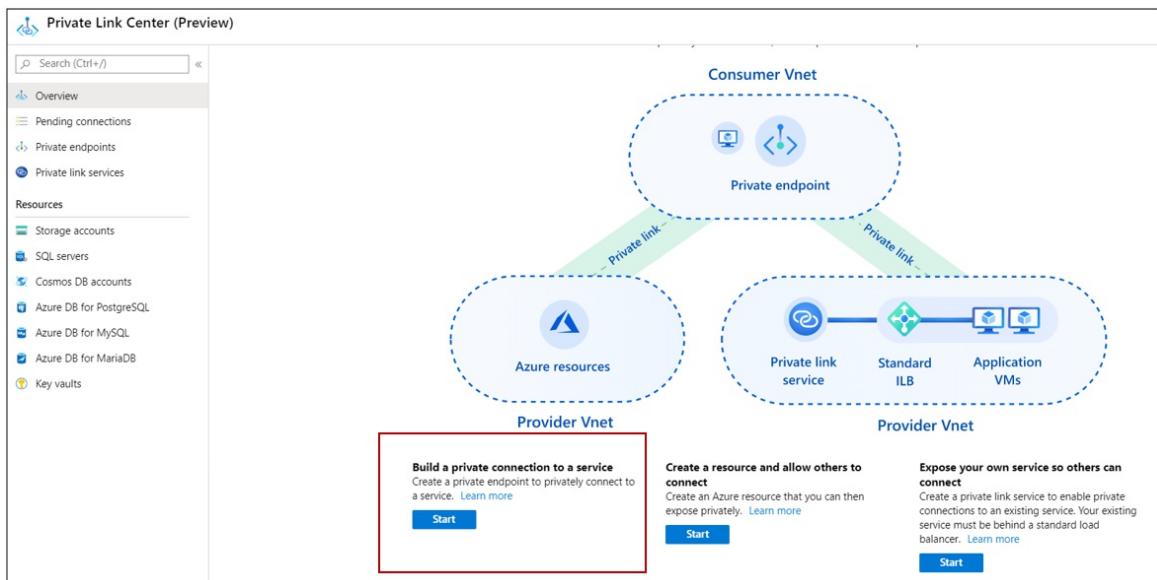
CONFIGURAÇÃO	VALOR
Subscription	Selecione sua assinatura.
Resource group	Selecione myResourceGroup . Você o criou na seção anterior.
Detalhes do servidor	
Nome do servidor	Insira <i>myserver</i> . Se esse nome já estiver sendo usado, crie um nome exclusivo.
Nome de usuário do administrador	Insira um nome de administrador de sua escolha.
Senha	Insira uma senha de sua escolha. A senha deve ter no mínimo 8 caracteres e atender a requisitos complexidade definidos.
Location	Selecione uma região do Azure na qual deseja que o PostgreSQL Server resida.
Versão	Selecione a versão do banco de dados do servidor PostgreSQL necessária.
Computação + armazenamento	Selecione o tipo de preço necessário para o servidor com base na carga de trabalho.

3. Selecione **OK**.
4. Selecione **Examinar + criar**. Você é levado até a página **Examinar + criar**, na qual o Azure valida sua configuração.
5. Quando a mensagem Validação aprovada for exibida, selecione **Criar**.
6. Quando vir a mensagem Validação aprovada, selecione **Criar**.

Criar um ponto de extremidade privado

Nesta seção, você criará um PostgreSQL Server e adicionará um ponto de extremidade privado a ele.

1. No lado superior esquerdo da tela no portal do Azure, selecione **Criar um recurso > Rede > Link Privado**.
2. Em **Central de Link Privado – Visão Geral**, na opção **Criar uma conexão privada com um serviço**, selecione **Iniciar**.



3. Em **Criar um ponto de extremidade privado – Noções básicas**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
Detalhes do projeto	
Subscription	Selecione sua assinatura.
Resource group	Selecione myResourceGroup . Você o criou na seção anterior.
Detalhes da Instância	
Nome	Insira <i>myPrivateEndpoint</i> . Se esse nome já estiver sendo usado, crie um nome exclusivo.
Região	Selecione Europa Ocidental .

4. Selecione **Avançar: Recurso**.

5. Em **Criar um ponto de extremidade privado – Recurso**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
Método de conexão	Selecionar conectar-se a um recurso do Azure em meu diretório.
Subscription	Selecione sua assinatura.
Tipo de recurso	Selecionar Microsoft.DBforPostgreSQL/servers .
Recurso	Selecionar <i>myServer</i>
Sub-recurso de destino	Selecionar <i>postgresqlServer</i>

6. Selecione **Avançar: configuração**.
7. Em **Criar um ponto de extremidade privado – configuração**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
REDE	
Rede virtual	Selecione <i>MyVirtualNetwork</i> .
Sub-rede	Selecione <i>mySubnet</i> .
INTEGRAÇÃO DE DNS PRIVADO	
Integrar com a zona DNS privado	Selecione Sim na barra superior.
Zona DNS privado	Selecione <i>(New)privatelink.postgres.database.azure.com</i>

NOTE

Use a zona DNS privada predefinida para seu serviço ou informe seu nome de zona DNS preferencial. Veja a [Configuração da zona DNS dos serviços do Azure](#) para obter detalhes.

8. Selecione **Examinar + criar**. Você é levado até a página **Examinar + criar**, na qual o Azure valida sua configuração.
9. Quando vir a mensagem **Validação aprovada**, selecione **Criar**.

Custom DNS settings	
FQDN	Private IP
demoprivatelinkserver.postgres.database.azure.com	10.1.3.4

NOTE

O FQDN na configuração de DNS do cliente não é resolvido para o IP privado configurado. Você precisará configurar uma zona DNS para o FQDN configurado, conforme mostrado [aqui](#).

Conectar-se a uma VM usando a RDP (Área de Trabalho Remota)

Depois de criar **myVm**, conecte-se a ela pela Internet da seguinte maneira:

1. Na barra de pesquisa do portal, insira *myVm*.
2. Selecione o botão **Conectar**. Depois de selecionar o botão **Conectar**, **Conectar-se à máquina virtual** abre.

3. Selecione **Baixar Arquivo RDP**. O Azure cria um arquivo *.rdp* (protocolo RDP) e ele é baixado no computador.

4. Abra o arquivo *downloaded.rdp*.

a. Se solicitado, selecione **Conectar**.

b. Insira o nome de usuário e a senha que você especificou ao criar a VM.

NOTE

Talvez seja necessário selecionar **Mais escolhas > Usar uma conta diferente** para especificar as credenciais inseridas durante a criação da VM.

5. Selecione **OK**.

6. Você pode receber um aviso do certificado durante o processo de logon. Se você receber um aviso de certificado, selecione **Sim** ou **Continuar**.

7. Depois que a área de trabalho da VM for exibida, minimize-a para voltar para sua área de trabalho local.

Acessar o servidor PostgreSQL de maneira privada a partir da VM

1. Na Área de Trabalho Remota do *myVM*, abra o PowerShell.

2. Insira `nslookup mydemopostgresserver.privatelink.postgres.database.azure.com`.

Você receberá uma mensagem semelhante a esta:

```
Server: Unknown
Address: 168.63.129.16
Non-authoritative answer:
Name: mydemopostgresserver.privatelink.postgres.database.azure.com
Address: 10.1.3.4
```

3. Teste a conexão de link particular para o servidor PostgreSQL usando qualquer cliente disponível. No exemplo abaixo, usei o [Azure Data Studio](#) para realizar a operação.

4. Em **Nova conexão**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
Tipo de servidor	Selecione PostgreSQL .
Nome do servidor	Selecione <i>mydemopostgresserver.privatelink.postgres.database.azure.com</i>
Nome de usuário	Insira o nome de usuário como <code>username@servername</code> , que é fornecido durante a criação do servidor PostgreSQL.
Senha	Insira uma senha fornecida durante a criação do servidor PostgreSQL.
SSL	Selecione Obrigatório .

5. Selecione Conectar.
6. Procurar bancos de dados no menu à esquerda.
7. (Opcionalmente) Crie ou consulte informações do servidor PostgreSQL.
8. Feche a conexão da área de trabalho remota com myVM.

Limpar recursos

Quando terminar de usar o ponto de extremidade privado, o servidor PostgreSQL e a VM, exclua o grupo de recursos e todos os recursos que ele contém:

1. Insira *myResourceGroup* na opção **Pesquisar** na parte superior do portal e selecione *myResourceGroup* nos resultados da pesquisa.
2. Selecione **Excluir grupo de recursos**.
3. Insira *myResourceGroup* em **DIGITAR O NOME DO GRUPO DE RECURSOS** e selecione **Excluir**.

Próximas etapas

Neste "como fazer", você criou uma VM em uma rede virtual, um Banco de Dados do Azure para PostgreSQL – Servidor único e um ponto de extremidade privado para acesso privado. Você se conectou a uma VM pela Internet e se comunicou com segurança com servidor PostgreSQL usando o Link Privado. Para saber mais sobre pontos de extremidade privados, confira [O que é o ponto de extremidade privado do Azure?](#).

Criar e gerenciar o Link Privado para o Banco de Dados do Azure para PostgreSQL – Servidor Único usando a CLI

21/05/2021 • 6 minutes to read

Um ponto de extremidade privado é o bloco de construção fundamental para o link privado no Azure. Ele permite que os recursos do Azure, como VMs (máquinas virtuais), se comuniquem de forma privada com recursos de link privado. Neste artigo, você aprenderá a usar o CLI do Azure para criar uma VM em uma rede virtual do Azure e um Servidor Único do Banco de Dados do Azure para PostgreSQL com um ponto de extremidade privado do Azure.

NOTE

O recurso de link privado está disponível apenas para servidores do Bancos de Dados do Azure para PostgreSQL nas camadas de preços de Uso geral ou Otimizado para Memória. Assegure-se de que o servidor de banco de dados esteja em um desses níveis de preços.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Um [banco de dados e servidor do Banco de Dados do Azure para PostgreSQL](#).

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.

3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.

4. Pressione **Enter** para executar o código.

Se você optar por instalar e usar a CLI do Azure localmente, este guia de início rápido exigirá a versão 2.0.28 ou posterior da CLI do Azure. Execute `az --version` para localizar a versão instalada. Para informações sobre como instalar ou atualizar, confira [Instalar a CLI do Azure](#).

Criar um grupo de recursos

Antes de criar qualquer recurso, você deve criar um grupo de recursos para hospedar a Rede Virtual. Crie um grupo de recursos com `az group create`. Este exemplo cria um grupo de recursos chamado *myResourceGroup* no local *westeurope*:

```
az group create --name myResourceGroup --location westeurope
```

Criar uma rede virtual

Crie uma Rede Virtual com `az network vnet create`. O exemplo cria uma Rede Virtual padrão nomeada *myVirtualNetwork* com uma sub-rede nomeada *mySubnet*:

```
az network vnet create \
--name myVirtualNetwork \
--resource-group myResourceGroup \
--subnet-name mySubnet
```

Desabilitar políticas de ponto de extremidade privado de sub-rede

O Azure implanta recursos em uma sub-rede dentro de uma rede virtual. Portanto, você precisa criar ou atualizar a sub-rede para desativar as [políticas de rede](#) de ponto de extremidade privado. Atualize uma configuração de sub-rede denominada *mySubnet* com `az network vnet subnet update`:

```
az network vnet subnet update \
--name mySubnet \
--resource-group myResourceGroup \
--vnet-name myVirtualNetwork \
--disable-private-endpoint-network-policies true
```

Criar a VM

Crie uma VM com `az vm create`. Quando solicitado, forneça uma senha a ser usada como credencial de entrada para a VM. Este exemplo cria uma VM chamada *myVm*:

```
az vm create \
--resource-group myResourceGroup \
--name myVm \
--image Win2019Datacenter
```

Anote o Endereço IP Público da VM. Você usará esse endereço para conectar-se à VM pela Internet na próxima etapa.

Criar um Banco de Dados do Azure para PostgreSQL – Servidor único

Crie um Banco de Dados do Azure para PostgreSQL com o comando az postgres server create. Lembre-se de que o nome do seu Servidor PostgreSQL deve ser exclusivo no Azure. Substitua o valor do espaço reservado pelos seus próprios valores exclusivos que você usou acima:

```
# Create a server in the resource group
az postgres server create \
--name mydemoserver \
--resource-group myresourcegroup \
--location westeurope \
--admin-user mylogin \
--admin-password <server_admin_password> \
--sku-name GP_Gen5_2
```

Criar um Ponto de Extremidade Privado

Crie um ponto de extremidade privado para o servidor PostgreSQL lógico em sua Rede Virtual:

```
az network private-endpoint create \
--name myPrivateEndpoint \
--resource-group myResourceGroup \
--vnet-name myVirtualNetwork \
--subnet mySubnet \
--private-connection-resource-id $(az resource show -g myResourceGroup -n mydemoserver --resource-type
"Microsoft.DBforPostgreSQL/servers" --query "id" -o tsv) \
--group-id postgresqlServer \
--connection-name myConnection
```

Configurar a Zona DNS Privada

Crie uma Zona DNS Privada para o domínio do servidor PostgreSQL e crie um link de associação com a Rede Virtual.

```
az network private-dns zone create --resource-group myResourceGroup \
--name "privatelink.postgres.database.azure.com"
az network private-dns link vnet create --resource-group myResourceGroup \
--zone-name "privatelink.postgres.database.azure.com"\ \
--name MyDNSLink \
--virtual-network myVirtualNetwork \
--registration-enabled false

#Query for the network interface ID
networkInterfaceId=$(az network private-endpoint show --name myPrivateEndpoint --resource-group
myResourceGroup --query 'networkInterfaces[0].id' -o tsv)

az resource show --ids $networkInterfaceId --api-version 2019-04-01 -o json
# Copy the content for privateIPAddress and FQDN matching the Azure database for PostgreSQL name

#Create DNS records
az network private-dns record-set a create --name myserver --zone-name
privatelink.postgres.database.azure.com --resource-group myResourceGroup
az network private-dns record-set a add-record --record-set-name myserver --zone-name
privatelink.postgres.database.azure.com --resource-group myResourceGroup -a <Private IP Address>
```

NOTE

O FQDN na configuração de DNS do cliente não é resolvido para o IP privado configurado. Você precisará configurar uma zona DNS para o FQDN configurado, conforme mostrado [aqui](#).

NOTE

Em alguns casos, o Banco de Dados do Azure para PostgreSQL e a sub-rede da VNet estão em assinaturas diferentes. Nesses casos, você deve garantir as seguintes configurações:

- Verifique se as duas assinaturas têm o provedor de recursos **Microsoft.DBforPostgreSQL** registrado. Para obter mais informações, confira [provedores de recursos](#).

Conecte uma VM a partir da Internet

Conecte-se à VM *myVm* da Internet da seguinte forma:

1. Na barra de pesquisa do portal, insira *myVm*.
2. Selecione o botão **Conectar**. Depois de selecionar o botão **Conectar**, **Conectar-se à máquina virtual** abre.
3. Selecione **Baixar Arquivo RDP**. O Azure cria um arquivo *.rdp* (protocolo RDP) e ele é baixado no computador.
4. Abra o arquivo *downloaded.rdp*.
 - a. Se solicitado, selecione **Conectar**.
 - b. Insira o nome de usuário e a senha que você especificou ao criar a VM.

NOTE

Talvez seja necessário selecionar **Mais escolhas > Usar uma conta diferente** para especificar as credenciais inseridas durante a criação da VM.

5. Selecione **OK**.
6. Você pode receber um aviso do certificado durante o processo de logon. Se você receber um aviso de certificado, selecione **Sim** ou **Continuar**.
7. Depois que a área de trabalho da VM for exibida, minimize-a para voltar para sua área de trabalho local.

Acessar o servidor PostgreSQL de maneira privada a partir da VM

1. Na Área de Trabalho Remota do *myVM*, abra o PowerShell.
2. Digite `nslookup mydemopostgresserver.privatelink.postgres.database.azure.com`.

Você receberá uma mensagem semelhante a esta:

```
Server: UnKnown
Address: 168.63.129.16
Non-authoritative answer:
Name: mydemopostgresserver.privatelink.postgres.database.azure.com
Address: 10.1.3.4
```

3. Teste a conexão de link particular para o servidor PostgreSQL usando qualquer cliente disponível. O exemplo a seguir usa o [Azure Data Studio](#) para realizar a operação.

4. Em **Nova conexão**, insira ou selecione estas informações:

CONFIGURAÇÃO	VALOR
Tipo de servidor	Selecione PostgreSQL .
Nome do servidor	Selecione <i>mydemopostgresserverprivatelink.postgres.database.azure.com</i>
Nome de usuário	Insira o nome de usuário como <code>username@servername</code> , que é fornecido durante a criação do servidor PostgreSQL.
Senha	Insira uma senha fornecida durante a criação do servidor PostgreSQL.
SSL	Selecione Obrigatório .

5. Selecione Conectar.

6. Procurar bancos de dados no menu à esquerda.

7. (Opcionalmente) Crie ou consulte informações do servidor PostgreSQL.

8. Feche a conexão da área de trabalho remota com myVM.

Limpar os recursos

Quando não for mais necessário, você poderá usar az group delete para remover o grupo de recursos e todos os recursos que ele contém:

```
az group delete --name myResourceGroup --yes
```

Próximas etapas

- Saiba mais sobre [O que é o ponto de extremidade privado do Azure](#)

Criptografia de dados para um servidor único do Banco de Dados do Azure para PostgreSQL usando o portal do Azure

21/05/2021 • 3 minutes to read

Saiba como usar o portal do Azure para configurar e gerenciar a criptografia de dados para o servidor único do Banco de Dados do Azure para PostgreSQL.

Pré-requisitos para o CLI do Azure

- É necessário ter uma assinatura do Azure e ser um administrador nessa assinatura.
- No Azure Key Vault, crie um cofre de chaves e uma chave para usar como chave gerenciada pelo cliente.
- O cofre de chaves deve ter as seguintes propriedades para ser usado como chave gerenciada pelo cliente:
 - [Exclusão reversível](#)

```
az resource update --id $(az keyvault show --name <key_vault_name> -test -o tsv | awk '{print $1}') --set properties.enableSoftDelete=true
```

- [Limpeza protegida](#)

```
az keyvault update --name <key_vault_name> --resource-group <resource_group_name> --enable-purge-protection true
```

- A chave deve ter os seguintes atributos a serem usados como chave gerenciada pelo cliente:
 - Sem data de validade
 - Não desabilitado
 - Capaz de realizar operações do tipo obter, codificar e decodificar chave

Definir as permissões corretas para operações de chave

1. No Key Vault, selecione Políticas de acesso > Adicionar Política de Acesso.

The screenshot shows the 'Access policies' section of the Azure Key Vault settings. On the left, there's a sidebar with 'demobyokkeyvault - Access policies' and a 'Key vault' section. The main area has tabs for 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', and 'Events (preview)'. Under 'Settings', 'Access policies' is selected and highlighted with a red box. The 'Enable Access to:' section contains three checkboxes: 'Azure Virtual Machines for deployment', 'Azure Resource Manager for template deployment', and 'Azure Disk Encryption for volume encryption'. Below this is a red-bordered 'Add Access Policy' button. The 'Current Access Policies' table lists one policy for 'User1' under the 'USER' category. The table columns are: Name, Category, Email, Key Permissions, Secret Permissions, Certificate Permissions, and Action. The 'Action' column for User1 contains buttons for '9 selected', '7 selected', '15 selected', and 'Delete'.

2. Selecione **Permissões de chave** e, em seguida, **Obter, Codificar, Decodificar** e também a **Entidade de segurança**, que é o nome do servidor PostgreSQL. Se a entidade de segurança do servidor não estiver na lista de entidades de segurança existentes, você precisará registrá-la. Você será solicitado a registrar a entidade de segurança do servidor se, ao tentar configurar a criptografia de dados pela primeira vez, ela falhar.

3. Selecione **Salvar**.

Defina a criptografia de dados para o servidor único do Banco de Dados do Azure para PostgreSQL

1. No Banco de Dados do Azure para PostgreSQL, selecione **Criptografia de dados** para configurar a chave gerenciada pelo cliente.

2. Você pode selecionar um cofre de chaves e um par de chaves, ou então inserir um identificador de chave.

demoprivateelinkserver - Data encryption
Azure Database for PostgreSQL-server

Connection security
Connection strings
Server parameters
Replication
Active Directory admin
Pricing tier
Properties
Locks
Export template
Data encryption
Use customer managed key Yes
Key selection method
Key vault *
Key *
Azure Database for PostgreSQL server uses Get, Wrap Key, Unwrap Key permissions to access the selected key vault. These permissions are only used to access the key vault for DataEncryption. If needed, we will try granting these permissions on your behalf. [Learn More](#)

3. Selecione **Salvar**.

4. Para garantir que todos os arquivos (incluindo arquivos temporários) sejam totalmente criptografados, reinicie o servidor.

Usar a criptografia de dados para servidores de restauração ou de réplica

Depois que o servidor único do Banco de Dados do Azure para PostgreSQL é criptografado com uma chave gerenciada pelo cliente armazenada no Key Vault, qualquer cópia recém-criada do servidor também é criptografada. Você pode fazer essa nova cópia seja com uma operação de restauração local ou geográfica, seja com uma operação de réplica (local/entre regiões). Portanto, no caso de um servidor PostgreSQL criptografado, você pode usar as etapas a seguir para criar um servidor restaurado criptografado.

1. No servidor, selecione **Visão geral > Restaurar**.

Reset password **Restore** Delete Restart Feedback

Resource group (change) : MyResourceGroup
Status : Available
Location : East US
Subscription (change) : MySubscription
Subscription ID :
Tags (change) : Click here to add tags

Server name : demobyoklocal.postgres.database.azure.com
Admin username : user@demobyoklocal
PostgreSQL version : 10
Performance configuration : General Purpose, 2 vCore(s), 100 GB
SSL enforce status : ENABLED

Resource utilization (demobyoklocal)

1 hour 24 hours 7 days Aggregation type: Avg

Time	Utilization (%)
1 hour ago	~0.5%
1 hour ago	~1.5%
1 hour ago	~1.0%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	~0.8%
1 hour ago	~0.9%
1 hour ago	~0.8%
1 hour ago	~0.7%
1 hour ago	~0.6%
1 hour ago	~0.5%
1 hour ago	~0.4%
1 hour ago	~0.3%
1 hour ago	~0.2%
1 hour ago	~0.1%
1 hour ago	~0.0%
1 hour ago	~0.1%
1 hour ago	~0.2%
1 hour ago	~0.3%
1 hour ago	~0.4%
1 hour ago	~0.5%
1 hour ago	~0.6%
1 hour ago	~0.7%
1 hour ago	

The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'byokdemoserver'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (with Connection security, Connection strings, Server parameters, Replication highlighted with a red box), Active Directory admin, Pricing tier, Properties, Locks, and Export template. The main content area displays the 'Master' and 'Replicas' sections. The 'Master' section shows the server name 'byokdemoserver', pricing tier 'General Purpose, 2 vCore(s), 100 GB', and location 'East US'. The 'Replicas' section shows 'No results'.

2. Depois que a operação de restauração for concluída, o novo servidor criado será criptografado com a chave do servidor primário. No entanto, os recursos e as opções no servidor estarão desabilitados e o servidor não estará acessível. Isso impede a manipulação de dados, já que a identidade do novo servidor ainda não recebeu permissão para acessar o cofre de chaves.

The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'demolocalreplica'. The left sidebar contains navigation links for Connection security, Connection strings, Server parameters, Replication (highlighted with a red box), Active Directory admin, Pricing tier, Properties, Locks, and Export template. The main content area displays the server configuration. A warning message at the top states: 'This server needs to validate the keys for this restored/replica server. Please validate the key information for this server.' The server details include: Resource group 'MyresourceGroup', Status 'Inaccessible', Location 'East US', Subscription 'MySubscription', Subscription ID, Tags (with a link to 'Click here to add tags'), and Resource utilization for 'demolocalreplica' showing 100% usage over the last hour.

3. Para tornar o servidor acessível, revalide a chave no servidor restaurado. Selecione **Criptografia de dados** > **Revalidar chave**.

NOTE

A primeira tentativa de revalidação falhará, pois a entidade de serviço do novo servidor precisa receber acesso ao cofre de chaves. Para gerar a entidade de serviço, selecione **Revalidar chave**, o que mostrará um erro, mas gerará a entidade de serviço. Depois disso, consulte [estas etapas](#) exibidas anteriormente neste artigo.

demolocalreplica - Data encryption
Azure Database for PostgreSQL server

Search (Ctrl+ /) Save Discard

Data encryption

Customer managed key encryption protects your database, backup, and logs using the user specified key without any change to your applications. To enable this protection, visit 'Data encryption' for each Azure Database for PostgreSQL server. [Learn More](#)

Use customer managed key

Step 1
Revalidate your existing key, or select a backup key.

Key selection method

Key identifier

Step 2
Attempt revalidation of your existing key. If successful, restoration of access to your data may take some time.

Você precisará conceder acesso ao cofre de chaves para o novo servidor.

4. Depois de registrar a entidade de serviço, re valide a chave mais uma vez, e o servidor retomará a funcionalidade normal.

demolocalreplica
Azure Database for PostgreSQL server

Search (Ctrl+ /) Delete Restart Feedback

Overview

Resource group (change) : MyResourceGroup
Status : Available
Location : East US
Subscription (change) : MySubscription
Subscription ID :
Tags (change) : Click here to add tags

Server name : demolocalreplica.postgres.database.azure.com
Admin username : user@demolocalreplica
PostgreSQL version : 10
Performance configuration : General Purpose, 2 vCore(s), 100 GB
SSL enforce status : ENABLED

Resource utilization (demolocalreplica)

1 hour 24 hours 7 days Aggregation type: Avg

CPU percent (Avg) 0.82% Storage percent (Avg) 0.47%

Notifications (0) Features (4) Tasks (1)

Próximas etapas

Para saber mais sobre a criptografia de dados, confira [Criptografia de dados de servidor único do Banco de Dados do Azure para PostgreSQL com chave gerenciada pelo cliente](#).

Criptografia de dados para um servidor único do Banco de Dados do Azure para PostgreSQL usando o CLI do Azure

21/05/2021 • 5 minutes to read

Saiba como usar a CLI do Azure para configurar e gerenciar a criptografia de dados para o servidor único do Banco de Dados do Azure para PostgreSQL.

Pré-requisitos para a CLI do Azure

- É necessário ter uma assinatura do Azure e ser um administrador nessa assinatura.
- Crie um cofre de chaves e uma chave para usar em uma chave gerenciada pelo cliente. Habilite também a proteção de limpeza e a exclusão temporária no cofre de chaves.

```
az keyvault create -g <resource_group> -n <vault_name> --enable-soft-delete true --enable-purge-protection true
```

- No Azure Key Vault criado, crie a chave que será usada para a criptografia de dados do servidor único do Banco de Dados do Azure para PostgreSQL.

```
az keyvault key create --name <key_name> -p software --vault-name <vault_name>
```

- Para usar um cofre de chaves existente, ele deve ter as seguintes propriedades para usar como uma chave gerenciada pelo cliente:

- Exclusão reversível

```
az resource update --id $(az keyvault show --name \ <key_vault_name> -o tsv | awk '{print $1}') --set \ properties.enableSoftDelete=true
```

- Limpeza protegida

```
az keyvault update --name <key_vault_name> --resource-group <resource_group_name> --enable-purge-protection true
```

- A chave deve ter os seguintes atributos a serem usados como chave gerenciada pelo cliente:

- Sem data de validade
 - Não desabilitado
 - Executar operações get, wrap e unwrap

Definir as permissões corretas para operações de chave

1. Há duas maneiras de obter a identidade gerenciada para o servidor único do Banco de Dados do Azure para PostgreSQL.

Crie um servidor de Banco de Dados do Azure para PostgreSQL com uma identidade gerenciada.

```
az postgres server create --name <server_name> -g <resource_group> --location <location> --storage-size <size> -u <user> -p <pwd> --backup-retention <7> --sku-name <sku name> --geo-redundant-backup <Enabled/Disabled> --assign-identity
```

Atualize um servidor do Banco de Dados do Azure para PostgreSQL para obter uma identidade gerenciada.

```
az postgres server update --resource-group <resource_group> --name <server_name> --assign-identity
```

2. Defina as **Permissões de chave (Get, Wrap, Unwrap)** para a **Entidade de Segurança**, que é o nome do servidor único do PostgreSQL.

```
az keyvault set-policy --name -g <resource_group> --key-permissions get unwrapKey wrapKey --object-id <principal id of the server>
```

Defina a criptografia de dados para o servidor único do Banco de Dados do Azure para PostgreSQL

1. Habilite a criptografia de dados para o servidor único do Banco de Dados do Azure para PostgreSQL usando a chave criada no Azure Key Vault.

```
az postgres server key create --name <server_name> -g <resource_group> --kid <key_url>
```

URL da chave:

<https://YourVaultName.vault.azure.net/keys/YourKeyName/01234567890123456789012345678901>

Usar a criptografia de dados para servidores de restauração ou de réplica

Depois que o servidor único do Banco de Dados do Azure para PostgreSQL é criptografado com uma chave gerenciada pelo cliente armazenada no Key Vault, qualquer cópia recém-criada do servidor também é criptografada. Você pode fazer essa nova cópia por meio de uma operação de restauração local ou geográfica ou por meio de réplicas de leitura (local/entre regiões). Portanto, para um servidor único do PostgreSQL criptografado, você pode usar as etapas a seguir para criar um servidor restaurado criptografado.

Criar um servidor restaurado/de réplica

- [Criar um servidor de restauração](#)
- [Criar um servidor de réplica de leitura](#)

Depois que o servidor for restaurado, revalidar a criptografia de dados do servidor restaurado

- Atribua a identidade para o servidor de réplica

```
az postgres server update --name <server_name> -g <resource_group> --assign-identity
```

- Obtenha a chave existente que deve ser usada para o servidor restaurado/de réplica

```
az postgres server key list --name '<server_name>' -g '<resource_group_name>'
```

- Defina a política para a nova identidade para o servidor restaurado/de réplica

```
az keyvault set-policy --name <keyvault> -g <resource_group> --key-permissions get unwrapKey wrapKey --object-id <principal id of the server returned by the step 1>
```

- Valide novamente o servidor restaurado/de réplica com a chave de criptografia

```
az postgres server key create -name <server name> -g <resource_group> --kid <key url>
```

Funcionalidade adicional para a chave que está sendo usada para o servidor único do Banco de Dados do Azure para PostgreSQL

Obter a chave usada

```
az postgres server key show --name <server name> -g <resource_group> --kid <key url>
```

URL da chave: <https://YourVaultName.vault.azure.net/keys/YourKeyName/01234567890123456789012345678901>

Listar a chave usada

```
az postgres server key list --name <server name> -g <resource_group>
```

Remova a chave que está sendo usada

```
az postgres server key delete -g <resource_group> --kid <key url>
```

Usar um modelo do Azure Resource Manager para habilitar a criptografia de dados

Além do portal do Azure, você também pode habilitar a criptografia de dados no servidor único do Banco de Dados do Azure para PostgreSQL usando modelos do Azure Resource Manager para servidores novos e existentes.

Para um servidor novo

Use um dos modelos do Azure Resource Manager criados previamente para provisionar o servidor com a criptografia de dados habilitada: [exemplo com criptografia de dados](#)

Este modelo do Azure Resource Manager cria um servidor único do Banco de Dados do Azure para PostgreSQL e usa o **KeyVault** e a **Chave** passados como parâmetros para habilitar a criptografia de dados no servidor.

Para um servidor existente

Além disso, você pode usar os modelos do Azure Resource Manager para habilitar a criptografia de dados em seus servidores únicos existentes do Banco de Dados do Azure para PostgreSQL.

- Passe a ID do recurso da chave do Azure Key Vault que você copiou anteriormente sob a propriedade `Uri` no objeto de propriedades.
- Use `2020-01-01-preview` como a versão da API.

```
{
  "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
```

```

    "type": "string"
},
"serverName": {
    "type": "string"
},
"keyVaultName": {
    "type": "string",
    "metadata": {
        "description": "Key vault name where the key to use is stored"
    }
},
"keyVaultResourceGroupName": {
    "type": "string",
    "metadata": {
        "description": "Key vault resource group name where it is stored"
    }
},
"keyName": {
    "type": "string",
    "metadata": {
        "description": "Key name in the key vault to use as encryption protector"
    }
},
"keyVersion": {
    "type": "string",
    "metadata": {
        "description": "Version of the key in the key vault to use as encryption protector"
    }
},
"variables": {
    "serverKeyName": "[concat(parameters('keyVaultName'), '_', parameters('keyName'), '_', parameters('keyVersion'))]"
},
"resources": [
{
    "type": "Microsoft.DBforPostgreSQL/servers",
    "apiVersion": "2017-12-01",
    "kind": "",
    "location": "[parameters('location')]",
    "identity": {
        "type": "SystemAssigned"
    },
    "name": "[parameters('serverName')]",
    "properties": {
    }
},
{
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2019-05-01",
    "name": "addAccessPolicy",
    "resourceGroup": "[parameters('keyVaultResourceGroupName')]",
    "dependsOn": [
        "[resourceId('Microsoft.DBforPostgreSQL/servers', parameters('serverName'))]"
    ],
    "properties": {
        "mode": "Incremental",
        "template": {
            "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
            "contentVersion": "1.0.0.0",
            "resources": [
                {
                    "type": "Microsoft.KeyVault/vaults/accessPolicies",
                    "name": "[concat(parameters('keyVaultName'), '/add')]",
                    "apiVersion": "2018-02-14-preview",
                    "properties": {
                        "accessPolicies": [
                            {
                                "tenantId": "[subscription().tenantId]"
                            }
                        ]
                    }
                }
            ]
        }
    }
}
]
}

```

```
        "objectId": "[reference(resourceId('Microsoft.DBforPostgreSQL/servers/'), parameters('serverName')), '2017-12-01', 'Full').identity.principalId]",
        "permissions": [
            "keys": [
                "get",
                "wrapKey",
                "unwrapKey"
            ]
        ]
    }
}
],
{
    "name": "[concat(parameters('serverName'), '/', variables('serverKeyName'))]",
    "type": "Microsoft.DBforPostgreSQL/servers/keys",
    "apiVersion": "2020-01-01-preview",
    "dependsOn": [
        "addAccessPolicy",
        "[resourceId('Microsoft.DBforPostgreSQL/servers', parameters('serverName'))]"
    ],
    "properties": {
        "serverKeyType": "AzureKeyVault",
        "uri": "[concat(reference(resourceId(parameters('keyVaultResourceGroupName'), 'Microsoft.KeyVault/vaults/'), parameters('keyVaultName')), '2018-02-14-preview', 'Full').properties.vaultUri, 'keys/', parameters('keyName'), '/', parameters('keyVersion'))]"
    }
}
]
```

Próximas etapas

Para saber mais sobre a criptografia de dados, confira [Criptografia de dados de servidor único do Banco de Dados do Azure para PostgreSQL com chave gerenciada pelo cliente](#).

Como validar a criptografia de dados do Banco de Dados do Azure para PostgreSQL

21/05/2021 • 2 minutes to read

Este artigo ajuda você a validar que a criptografia de dados usando a chave gerenciada pelo cliente para o Banco de Dados do Azure para PostgreSQL está funcionando conforme o esperado.

Verificar o status de criptografia

No portal

1. Se você quiser verificar se a chave do cliente é usada para criptografia, siga estas etapas:

- No portal do Azure, navegue até as **Azure Key Vault -> Chaves**
- Selecione a chave usada para criptografia do servidor.
- Defina o status da chave como **Habilitada** como **Não**.

Após algum tempo (~15 min), o **status** do servidor de Banco de Dados do Azure para PostgreSQL deve estar **Inacessível**. Qualquer operação de E/S feita no servidor falhará, o que validará que o servidor está realmente criptografado com a chave de clientes e que a chave não é válida no momento.

Para **Disponibilizar** o servidor com relação a ela, você pode revalidar a chave.

- Defina o status da chave no cofre de chaves como **Sim**.
- Em **Criptografia de Dados**, selecione **Revalidar chave**.
- Depois que a revalidação da chave for bem-sucedida, o **status** do servidor mudará para **Disponível**

2. No portal do Azure, se você puder garantir que a chave de criptografia esteja definida, os dados serão criptografados usando a chave de clientes usada no portal do Azure.

The screenshot shows the 'Data encryption' settings for the 'demopostgresql1' PostgreSQL server. The 'Use customer managed key' section is highlighted with a red box. It shows 'Yes' selected for 'Use customer managed key' and a key identifier URL 'https://mysqldemovault.vault.azure.net/keys/tempkey/402c5cd7d13c4118b05d1...'. A note at the bottom explains that the database uses Get, Wrap Key, Unwrap Key permissions to access the selected key vault.

Da CLI

1. Podemos usar o comando `az CL` para validar os principais recursos que estão sendo usados para o servidor do Banco de Dados do Azure para PostgreSQL.

```
az postgres server key list --name '<server_name>' -g '<resource_group_name>'
```

Para um servidor sem criptografia de dados definida, esse comando resultará em um conjunto vazio [].

Relatórios de auditoria do Azure

Os [Relatórios de Auditoria](#) também podem ser analisados, fornecendo informações sobre a conformidade com os padrões de proteção de dados e os requisitos regulatórios.

Próximas etapas

Para saber mais sobre a criptografia de dados, confira [Criptografia de dados de servidor único do Banco de Dados do Azure para PostgreSQL com chave gerenciada pelo cliente](#).

Criptografia dupla de infraestrutura para o banco de dados do Azure para PostgreSQL

21/05/2021 • 2 minutes to read

Saiba como usar o como configurar e gerenciar a criptografia dupla de infraestrutura para o banco de dados do Azure para PostgreSQL.

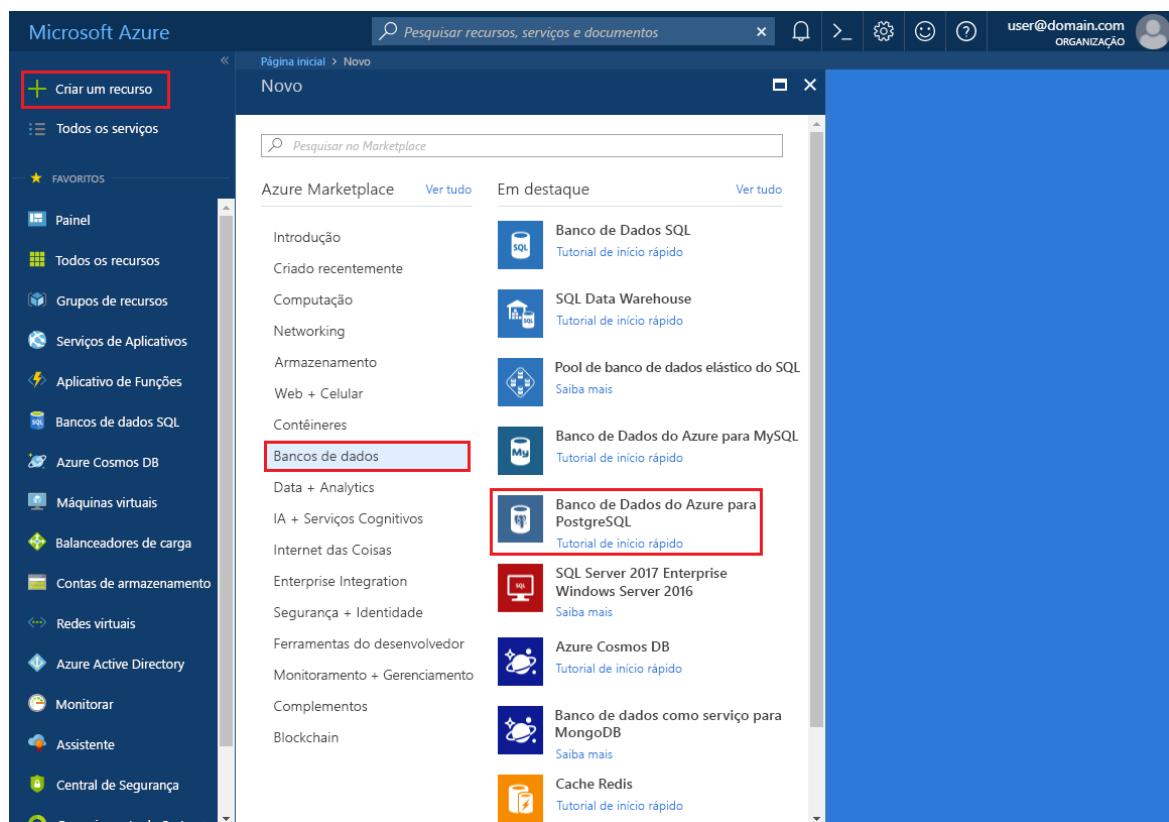
Pré-requisitos

- É necessário ter uma assinatura do Azure e ser um administrador nessa assinatura.

Criar um servidor de banco de dados do Azure para PostgreSQL com criptografia dupla de infraestrutura-Portal

Siga estas etapas para criar um servidor de banco de dados do Azure para PostgreSQL com criptografia dupla de infraestrutura do portal do Azure:

1. Selecione **Criar um recurso** (+) no canto superior esquerdo do portal.
2. Selecione **Bancos de Dados > Banco de Dados do Azure para PostgreSQL**. Você também pode inserir PostgreSQL na caixa de pesquisa para localizar o serviço. Habilitada a opção de implantação de **servidor único**.



3. Forneça as informações básicas do servidor. Selecione **configurações adicionais** e habilite a caixa de seleção **criptografia dupla de infraestrutura** para definir o parâmetro.

Single server

Microsoft

Basics Additional settings Tags Review + create

Customize additional configuration parameters for database server.

Data Security

Protect data using additional controls for your database server.

Infrastructure (Double) encryption ⓘ Infrastructure encryption enabled

Review + create

< Previous

Next : Tags >

4. Selecione Examinar + criar para provisionar o servidor.

Single server

Microsoft

Basics Additional settings Tags Review + create

Product details

Azure Database for PostgreSQL
by Microsoft

[Terms of use](#) | [Privacy policy](#)

Estimated cost per month

141.85 USD

[View pricing details](#)

Terms

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft third-party offerings. For additional details see [Azure Marketplace Terms](#).

Basics

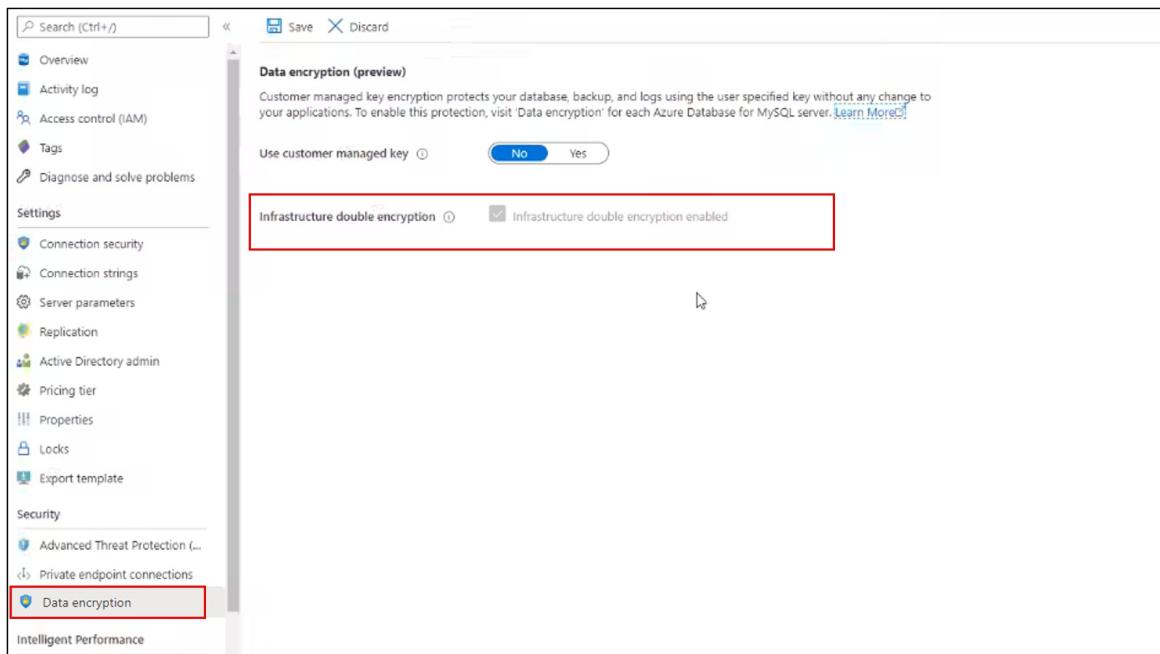
Subscription	Orcas PM team
Resource group	manishku
Server name	mydemoserverinfrapg
Data source	None
Server admin login name	manishthe
Location	East US
Version	10
Compute + storage	GeneralPurpose, Gen5, 2 vCores, 100 GB Storage
Backup retention period	7 day(s)
Backup redundancy	Locally redundant
Storage Auto Grow	Enabled
Infrastructure (Double) encryption	Enabled

Create

< Previous

[Download a template for automation](#)

5. Depois que o servidor for criado, você poderá validar a criptografia dupla da infraestrutura verificando o status na folha do servidor de criptografia de dados .



Criar um servidor de banco de dados do Azure para PostgreSQL com criptografia dupla de infraestrutura-CLI

Siga estas etapas para criar um servidor de banco de dados do Azure para PostgreSQL com criptografia dupla de infraestrutura da CLI:

Este exemplo cria um grupo de recursos chamado `myresourcegroup` no `westus` local.

```
az group create --name myresourcegroup --location westus
```

O exemplo a seguir cria um servidor PostgreSQL 11 no oeste dos EUA chamado `mydemoserver` em seu grupo de recursos `myresourcegroup` com logon de administrador do servidor `myadmin`. Esse é um servidor **Gen 4 de Uso Geral** com 2 vCores. Isso também habilitará a criptografia dupla de infraestrutura para o servidor criado. Substitua o `<server_admin_password>` com seu próprio valor.

```
az postgres server create --resource-group myresourcegroup --name mydemoserver --location westus --admin-user myadmin --admin-password <server_admin_password> --sku-name GP_Gen4_2 --version 11 --infrastructure-encryption >Enabled/Disabled>
```

Próximas etapas

Para saber mais sobre criptografia de dados, confira [criptografia dupla de banco de dados do Azure para PostgreSQL](#).

Negar Acesso à Rede Pública no Servidor único do Banco de Dados do Azure para PostgreSQL usando o portal do Azure

21/05/2021 • 2 minutes to read

Este artigo descreve como você pode configurar um Servidor único do Banco de Dados do Azure para PostgreSQL para negar todas as configurações públicas e permitir somente conexões por meio de pontos de extremidade privados para aprimorar ainda mais a segurança da rede.

Pré-requisitos

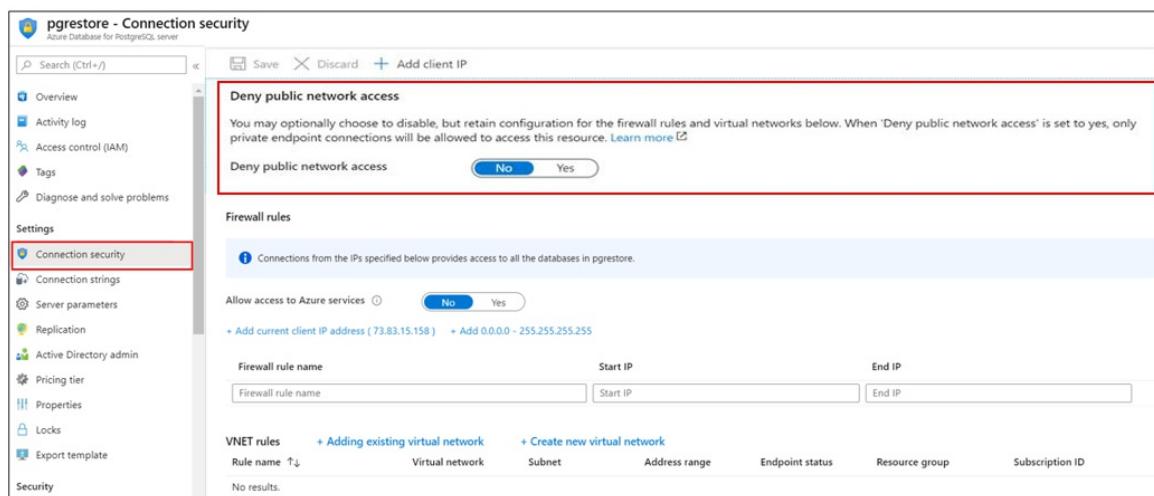
Para concluir este guia de instruções, você precisa:

- Um [Servidor único do Banco de Dados do Azure para PostgreSQL](#) com tipo de preço Uso Geral ou Otimizado para Memória.

Configurar o Negar Acesso à Rede Pública

Siga estas etapas para configurar o Negar Acesso à Rede Pública do Servidor único do PostgreSQL:

1. Na [portal do Azure](#), selecione o atual Servidor único do Banco de Dados do Azure para PostgreSQL.
2. Na página do Servidor único do PostgreSQL, em **Configurações**, clique em **Segurança de conexão** para abrir a página configuração de segurança de conexão.
3. Em **Negar Acesso à Rede Pública**, selecione **Sim** para negar o acesso público do Servidor único do PostgreSQL.



4. Clique em **Salvar** para salvar as alterações.
5. Uma notificação confirmará que a configuração de segurança da conexão foi habilitada com êxito.

The screenshot shows the 'Connection security' blade for a PostgreSQL server named 'pgstore'. The left sidebar lists various settings like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Connection security (which is selected). The main area shows the 'Deny public network access' setting is set to 'Yes'. Below it, there's a section for Firewall rules and a note about allowing access to Azure services. A success message in the activity log states 'Successfully updated the connection security settings'.

Próximas etapas

Saiba mais sobre [como criar alertas sobre métricas](#).

Criar e gerenciar réplicas de leitura no Banco de Dados do Azure para PostgreSQL – Servidor Único usando o portal do Azure

21/05/2021 • 5 minutes to read

Neste artigo, você aprenderá a criar e gerenciar réplicas de leitura no Banco de Dados do Azure para PostgreSQL no portal do Azure. Para saber mais sobre réplicas de leitura, confira [Visão Geral](#).

Pré-requisitos

Um [servidor do Banco de Dados do Azure para PostgreSQL](#) que será o servidor primário.

Suporte para replicação do Azure

As [réplicas de leitura](#) e a [decodificação lógica](#) dependem do WAL (Log Write-Ahead) do Postgres para obter informações. Esses dois recursos precisam de diferentes níveis de registro em log do Postgres. A decodificação lógica precisa de um nível mais alto de registro em log do que as réplicas de leitura.

Para configurar o nível correto de registro em log, use o parâmetro de suporte à replicação do Azure. O suporte à replicação do Azure tem três opções de configuração:

- **Desativado** – coloca o mínimo de informações no WAL. Essa configuração não está disponível na maioria dos servidores do Banco de Dados do Azure para PostgreSQL.
- **Réplica** – mais detalhado do que **Desativado**. É o nível mínimo de registro em log necessário para que as [réplicas de leitura](#) funcionem. Essa configuração é o padrão na maioria dos servidores.
- **Lógico** – mais detalhado do que **Réplica**. É o nível mínimo de registro em log para que a decodificação lógica funcione. As réplicas de leitura também funcionam com essa configuração.

NOTE

Ao implantar réplicas de leitura para cargas de trabalho primárias persistentes, pesadas e com uso intensivo de gravação, o atraso de replicação pode continuar aumentando e nunca alcançar a primária. Isso também pode aumentar o uso de armazenamento na primária, pois os arquivos do WAL não são excluídos até que sejam recebidos na réplica.

Preparar o servidor primário

1. No portal do Azure, selecione um servidor do Banco de Dados do Azure para PostgreSQL existente para ser usado como mestre.
2. No menu do servidor, selecione **Replicação**. Se o suporte à replicação do Azure estiver definido como, no mínimo, **Réplica**, você poderá criar réplicas de leitura.
3. Se o suporte à replicação do Azure não estiver definido como, no mínimo, **Réplica**, defina-o. Selecione **Salvar**.

myserver | Replication

Azure Database for PostgreSQL server

Search (Ctrl+ /)

Add Replica | Delete Replica | Stop Replication | Save | Discard

Azure replication support [Learn more](#) OFF REPLICA LOGICAL

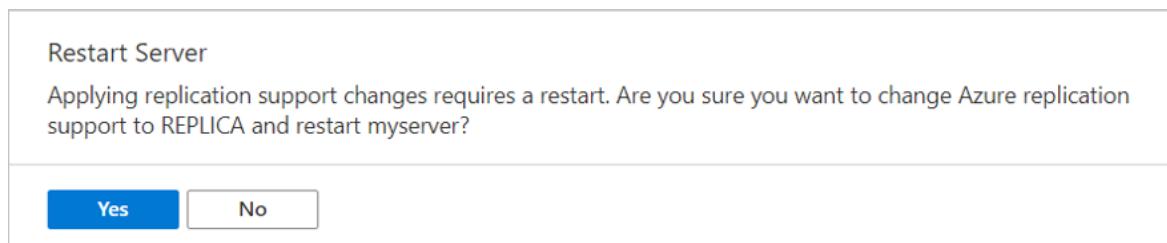
Master

Name	Pricing tier	Location	Status
myserver	General Purpose, 2 vCore(s), ...	East US	Available

Replicas

Name	Pricing tier	Location	Status
No results			

4. Reinicie o servidor para aplicar a alteração selecionando Sim.



5. Você receberá duas notificações do portal do Azure quando a operação for concluída. Há uma notificação para atualizar o parâmetro do servidor. Logo em seguida, há outra notificação para a reinicialização do servidor.

Notifications

More events in the activity log → Dismiss all

Successfully restarted the PostgreSQL server 3 minutes ago

Successfully updated the server parameters 5 minutes ago

6. Atualize a página do portal do Azure para atualizar a barra de ferramentas de Replicação. Agora, você pode criar réplicas de leitura para esse servidor.

Criar uma réplica de leitura

Para criar uma réplica de leitura, siga estas etapas:

1. Selecione servidor do Banco de Dados do Azure para PostgreSQL existente a ser usado como servidor primário.
2. Na barra lateral do servidor, em **CONFIGURAÇÕES**, selecione **Replicação**.
3. Selecione **para adicionar réplica**.

The screenshot shows the Microsoft Azure portal interface. The URL in the address bar is `portal.azure.com`. The main title is "Replication - Microsoft Azure". The breadcrumb navigation shows "Home > mydemoserver - Replication". The left sidebar has a tree view with "mydemoserver - Replication" selected. Under "mydemoserver - Replication", there are several options: Overview, Activity log, Tags, Settings (with sub-options like Connection security, Connection strings, Server parameters, and Replication), Pricing tier, Properties, Locks, and Automation script. The "Replication" option is highlighted with a red box. On the right, there are two tables: "MASTER" and "REPLICAS". The "MASTER" table shows one entry: "mydemoserver" with a "Pricing tier" of "General Purpose, 4 vCore(s), 576 GB" and a "Location" of "Brazil South". The "REPLICAS" table shows "No results". At the top right, there are buttons for "Add Replica", "Delete Replica", and "Stop Replication". A search bar at the top says "Search resources, services, and docs".

4. Insira um nome para a réplica de leitura.

Home > mydemoserver - Replication > PostgreSQL server

PostgreSQL server

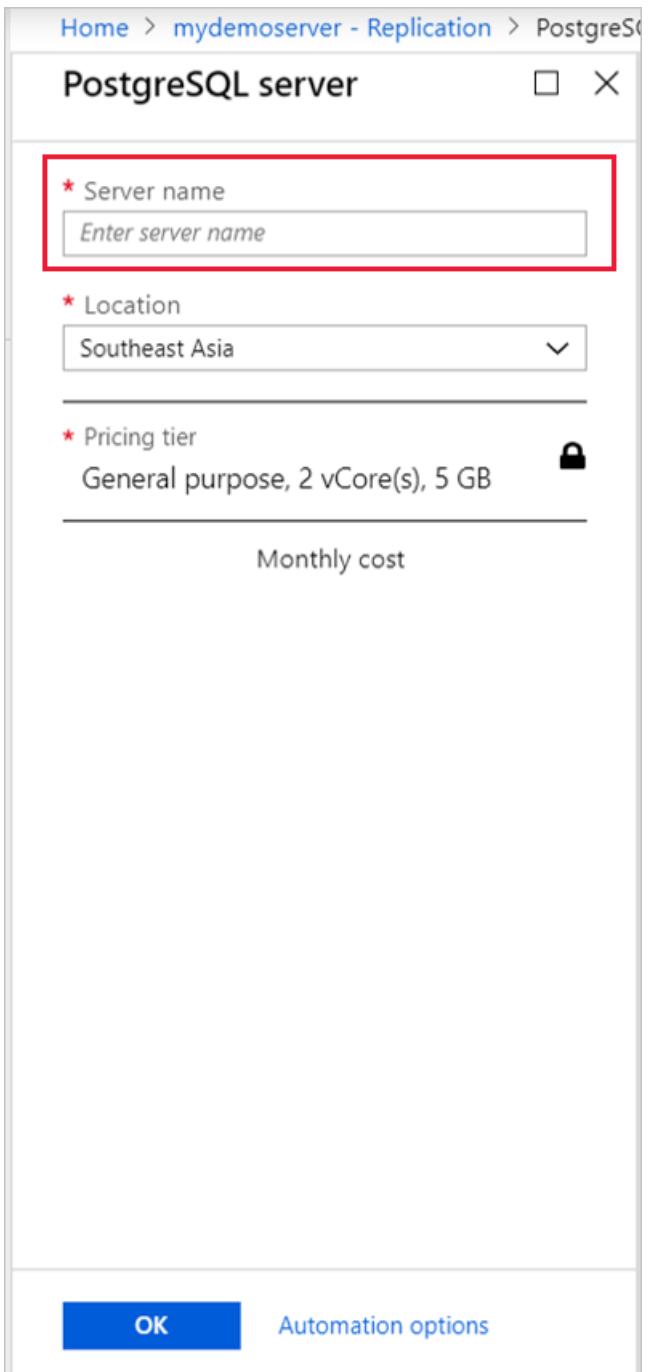
* Server name
Enter server name

* Location
Southeast Asia

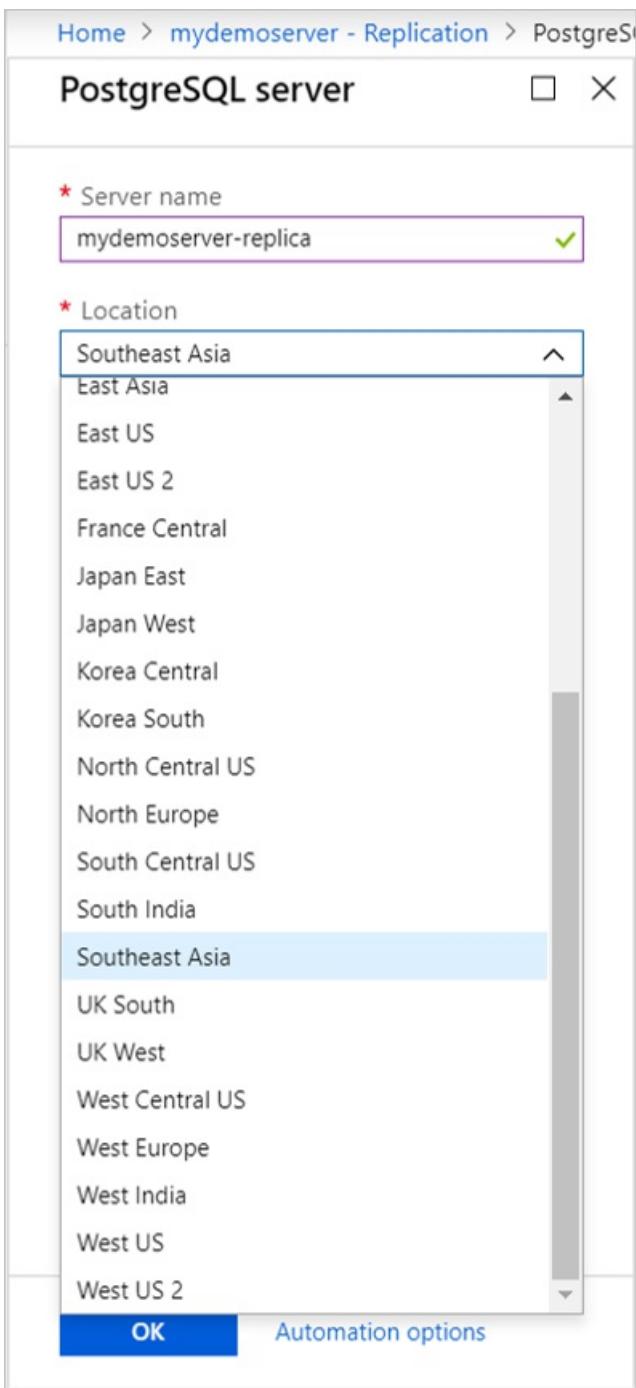
* Pricing tier
General purpose, 2 vCore(s), 5 GB 

Monthly cost

OK Automation options



5. Selecione um local para a réplica. O local padrão é o mesmo que o do servidor primário.



NOTE

Para saber mais sobre em quais regiões você pode criar uma réplica, visite o artigo [conceitos de réplica de leitura](#).

6. Selecione **OK** para confirmar a criação da réplica.

Depois que a réplica de leitura é criada, ela pode ser exibida na janela **Replicação**:

The screenshot shows the 'mydemoserver - Replication' page in the Azure portal. On the left, there's a sidebar with 'Overview', 'Activity log', 'Tags', 'Settings' (with 'Connection security', 'Connection strings', 'Server parameters', and 'Replication' selected), and a search bar. The main area has three tabs: 'Add Replica', 'Delete Replica', and 'Stop Replication'. Below these are two tables: 'MASTER' and 'REPLICAS'. The 'MASTER' table shows one entry: 'mydemoserver' with a 'General Purpose, 4 vCore(s), 576 GB' tier and 'Brazil South' location. The 'REPLICAS' table shows two entries: 'mydemoserver-replica' and 'mydemoserver-replica2', both with the same tier and location.

NAME	PRICING TIER	LOCATION
mydemoserver	General Purpose, 4 vCore(s), 576 GB	Brazil South

NAME	PRICING TIER	LOCATION
mydemoserver-replica	General Purpose, 4 vCore(s), 576 GB	Brazil South
mydemoserver-replica2	General Purpose, 4 vCore(s), 576 GB	Brazil South

IMPORTANT

Examine a [seção de considerações da Visão geral da réplica de leitura](#).

Antes que uma configuração do servidor primário seja atualizada para um novo valor, atualize a configuração de réplica para um valor igual ou maior. Isso ajuda a réplica a acompanhar as alterações feitas ao mestre.

Parar replicação

Você pode parar a replicação entre um servidor primário e uma réplica de leitura.

IMPORTANT

Após parar a replicação para um servidor primário e uma réplica de leitura, isso não poderá ser desfeito. A réplica de leitura se torna um servidor autônomo que dá suporte a leituras e gravações. O servidor autônomo não pode se tornar uma réplica novamente.

Para parar a replicação entre um servidor primário e uma réplica de leitura no portal do Azure, siga estas etapas:

1. No portal do Azure, selecione o servidor primário do Banco de Dados do Azure para PostgreSQL.
2. No menu de servidor, em **CONFIGURAÇÕES**, selecione **Replicação**.
3. Selecione o servidor de réplica para o qual interromper a replicação.

The screenshot shows the 'mydemoserver - Replication' page. The 'Stop Replication' button is highlighted with a red box. The 'REPLICAS' table shows three entries: 'mydemoserver-replica2', 'mydemoserver-replica', and 'mydemoserver'. The row for 'mydemoserver-replica' is also highlighted with a red box.

NAME	PRICING TIER	LOCATION
mydemoserver-replica2	General Purpose, 4 vCore(s), 576 GB	Brazil South
mydemoserver-replica	General Purpose, 4 vCore(s), 576 GB	Brazil South
mydemoserver	General Purpose, 4 vCore(s), 576 GB	Brazil South

4. Selecione **Parar replicação**.

The screenshot shows the 'General' tab of a PostgreSQL server's configuration page. At the top, there are three buttons: '+ Add Replica', 'Delete Replica', and 'Stop Replication' (which is highlighted with a red box). Below these are two sections: 'MASTER' and 'REPLICAS'. The 'MASTER' section lists one entry: 'mydemoserver' with a 'General Purpose, 4 vCore(s), 576 GB' tier and 'Brazil South' location. The 'REPLICAS' section lists two entries: 'mydemoserver-replica2' and 'mydemoserver-replica', both with the same specifications.

5. Selecione **OK** para interromper a replicação.

This is a confirmation dialog box. It contains the text: 'Stop Replication', 'Stopping replication to mydemoserver-replica is irreversible. The action you're about to take can't be undone. Going further will remove replication between this replica and the master.', and 'mydemoserver-replica will become a standalone server.' At the bottom are two buttons: 'OK' (highlighted with a red box) and 'Cancel'.

Excluir um servidor primário

Para excluir um servidor primário, use as mesmas etapas usadas para excluir um servidor autônomo do Banco de Dados do Azure para PostgreSQL.

IMPORTANT

Ao excluir um servidor primário, a replicação para todas as réplicas de leitura será interrompida. As réplicas de leitura tornam-se servidores independentes que agora têm suporte para leitura e gravação.

Para excluir um servidor do portal do Azure, siga estas etapas:

1. No portal do Azure, selecione o servidor primário do Banco de Dados do Azure para PostgreSQL.
2. Abra a página **Visão geral** para o servidor. Selecione **Excluir**.

The screenshot shows the Azure portal interface for managing a PostgreSQL server named 'mydemoserver'. The left sidebar has 'Overview' selected. The main content area displays 'Essentials' information including resource group ('myresourcegroup'), status ('Available'), location ('Brazil South'), subscription name ('mysubscription'), and subscription ID. A prominent 'Delete' button is highlighted with a red box at the top right of the essentials section.

3. Insira o nome do servidor primário a ser excluído. Selecione Excluir para confirmar a exclusão do servidor primário.

This screenshot shows a confirmation dialog box titled 'Are you sure you want to delete mydemoserver?'. It contains a warning message about the irreversibility of the action. A text input field labeled 'TYPE THE AZURE DATABASE FOR POSTGRESQL SERVER NAME' contains the value 'mydemoserver', which is also highlighted with a red box. At the bottom, there are two buttons: 'Delete' (highlighted with a red box) and 'Cancel'.

Excluir uma réplica

Você pode excluir uma réplica de leitura semelhante a como exclui um servidor primário.

- No portal do Azure, abra a página Visão geral para a réplica de leitura. Selecione Excluir.

Home > mydemoserver-replica

mydemoserver-replica
Azure Database for PostgreSQL server

Search (Ctrl+/
Overview Essentials

Reset password Restore Delete

Você também pode excluir a réplica de leitura usando a janela **Replicação** seguindo estas etapas:

1. No portal do Azure, selecione o servidor primário do Banco de Dados do Azure para PostgreSQL.
2. No menu de servidor, em **CONFIGURAÇÕES**, selecione **Replicação**.
3. Selecione a réplica de leitura a excluir.

+ Add Replica Delete Replica Stop Replication

MASTER

NAME	PRICING TIER	LOCATION
mydemoserver	General Purpose, 4 vCore(s), 576 GB	Brazil South

REPLICAS

NAME	PRICING TIER	LOCATION
mydemoserver-replica2	General Purpose, 4 vCore(s), 576 GB	Brazil South
mydemoserver-replica	General Purpose, 4 vCore(s), 576 GB	Brazil South

4. Selecione **Excluir réplica**.

+ Add Replica Delete Replica Stop Replication

MASTER

NAME	PRICING TIER	LOCATION
mydemoserver	General Purpose, 4 vCore(s), 576 GB	Brazil South

REPLICAS

NAME	PRICING TIER	LOCATION
mydemoserver-replica2	General Purpose, 4 vCore(s), 576 GB	Brazil South
mydemoserver-replica	General Purpose, 4 vCore(s), 576 GB	Brazil South

5. Insira o nome da réplica a excluir. Selecione **Excluir** para confirmar a exclusão da réplica.

Are you sure you want to delete mydemoserver-replica?



Warning! Deleting mydemoserver-replica is irreversible. The action you're about to take can't be undone. Going further will delete it and all the items in it permanently.

TYPE THE AZURE DATABASE FOR POSTGRESQL SERVER NAME

mydemoserver-replica

Affected items

DATABASE TO BE REMOVED

postgres

azure_maintenance

azure_sys

Monitorar uma réplica

Duas métricas estão disponíveis para monitorar réplicas de leitura.

Métrica de Retardo Máximo Entre Réplicas

A métrica **Retardo Máximo entre Réplicas** mostra o retardo em bytes entre o servidor primário e a réplica com o maior retardo.

1. No portal do Azure, selecione o servidor primário do Banco de Dados do Azure para PostgreSQL.
2. Selecione **Métricas**. Na janela **Métricas**, selecione **Retardo Máximo entre Réplicas**.

The screenshot shows the Azure portal's Metrics blade for a PostgreSQL server named 'mydemoserver-replica'. The left sidebar is filled with monitoring and security settings. The main area is titled 'New chart' and contains a dropdown menu for selecting a metric. The 'Max Lag Across Replicas' option is clearly highlighted with a red box.

3. Para sua Agregação, selecione Máx.

Métrica de retardo de réplica

A métrica **Retardo da Réplica** mostra o tempo decorrido desde a última transação reproduzida em uma réplica. Se não houver nenhuma transação ocorrendo no mestre, a métrica refletirá esse retardo.

1. No portal do Azure, selecione a réplica de leitura do Banco de Dados do Azure para PostgreSQL.

2. Selecione **Métricas**. Na janela **Métricas**, selecione **Retardo de Réplica**.

This screenshot is similar to the previous one but focuses on the 'Replica Lag (seconds)' metric. The dropdown menu in the 'New chart' section has 'Replica Lag (seconds)' highlighted with a red box.

3. Para sua Agregação, selecione Máx.

Próximas etapas

- Saiba mais sobre [réplicas de leitura no Banco de Dados do Azure para PostgreSQL](#).
- Saiba como [criar e gerenciar réplicas de leitura na CLI do Azure e na API REST](#).

Criar e gerenciar réplicas de leitura na CLI do Azure, API REST

25/05/2021 • 6 minutes to read

Neste artigo, você aprenderá a criar e gerenciar réplicas de leitura no Banco de Dados do Azure para PostgreSQL usando a CLI do Azure e API REST. Para saber mais sobre réplicas de leitura, confira [Visão Geral](#).

Supporte para replicação do Azure

As [réplicas de leitura](#) e a [decodificação lógica](#) dependem do WAL (Log Write-Ahead) do Postgres para obter informações. Esses dois recursos precisam de diferentes níveis de registro em log do Postgres. A decodificação lógica precisa de um nível mais alto de registro em log do que as réplicas de leitura.

Para configurar o nível correto de registro em log, use o parâmetro de suporte à replicação do Azure. O suporte à replicação do Azure tem três opções de configuração:

- **Desativado** – coloca o mínimo de informações no WAL. Essa configuração não está disponível na maioria dos servidores do Banco de Dados do Azure para PostgreSQL.
- **Réplica** – mais detalhado do que **Desativado**. É o nível mínimo de registro em log necessário para que as [réplicas de leitura](#) funcionem. Essa configuração é o padrão na maioria dos servidores.
- **Lógico** – mais detalhado do que **Réplica**. É o nível mínimo de registro em log para que a decodificação lógica funcione. As réplicas de leitura também funcionam com essa configuração.

NOTE

Ao implantar réplicas de leitura para cargas de trabalho primárias persistentes, pesadas e com uso intensivo de gravação, o atraso de replicação pode continuar aumentando e nunca alcançar a primária. Isso também pode aumentar o uso de armazenamento no primário, pois os arquivos WAL não são excluídos até que sejam recebidos na réplica.

CLI do Azure

Saiba como criar e gerenciar réplicas de leitura usando a CLI do Azure.

Pré-requisitos

- [Instalar a CLI 2.0 do Azure](#)
- Um [servidor do Banco de Dados do Azure para PostgreSQL](#) que será o servidor primário.

Preparar o servidor primário

1. Verifique o valor do servidor primário `azure.replication_support`. Ele deve ser pelo menos REPLICA para que réplicas de leitura funcionem.

```
az postgres server configuration show --resource-group myresourcegroup --server-name mydemoserver --name azure.replication_support
```

2. Se `azure.replication_support` não for pelo menos a REPLICA, defina-a.

```
az postgres server configuration set --resource-group myresourcegroup --server-name mydemoserver --name azure.replication_support --value REPLICA
```

3. Reinicie o servidor para aplicar a alteração.

```
az postgres server restart --name mydemoserver --resource-group myresourcegroup
```

Criar uma réplica de leitura

O comando `az postgres server replica create` requer os seguintes parâmetros:

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
resource-group	myresourcegroup	O grupo de recursos para o qual o servidor de réplica será criado.
name	mydemoserver-replica	O nome do novo servidor de réplica criado.
source-server	mydemoserver	O nome ou a ID do servidor de origem existente para replicar. Use a ID do recurso se desejar que os grupos de recursos da réplica e do mestre sejam diferentes.

No exemplo de CLI abaixo, a réplica é criada na mesma região que o mestre.

```
az postgres server replica create --name mydemoserver-replica --source-server mydemoserver --resource-group myresourcegroup
```

Para criar uma réplica de leitura entre regiões, use o parâmetro `--location`. O exemplo da CLI abaixo cria a réplica no Oeste dos EUA.

```
az postgres server replica create --name mydemoserver-replica --source-server mydemoserver --resource-group myresourcegroup --location westus
```

NOTE

Para saber mais sobre em quais regiões você pode criar uma réplica, visite o artigo [conceitos de réplica de leitura](#).

Se você não tiver definido o `azure.replication_support` parâmetro para **REPLICA** em um servidor primário Optimizado para Memória ou de Uso Geral e reiniciado o servidor, você receberá uma notificação de erro. Conclua essas duas etapas antes de criar uma réplica.

IMPORTANT

Examine a [seção considerações da Visão Geral da Réplica de leitura](#).

Antes que uma configuração do servidor primário seja atualizada para um novo valor, atualize a configuração de réplica para um valor igual ou maior. Esta ação ajuda a réplica a acompanhar as alterações feitas ao mestre.

Listar réplicas

Você pode exibir a lista de réplicas de um servidor primário usando o comando `az postgres server replica list`.

```
az postgres server replica list --server-name mydemoserver --resource-group myresourcegroup
```

Parar a replicação para um servidor de réplica

Você pode interromper a replicação entre um servidor primário e uma réplica de leitura usando o comando [az postgres server replica stop](#).

Depois de interromper a replicação para um servidor primário e uma réplica de leitura, isso não poderá ser desfeito. A réplica de leitura se torna um servidor autônomo que dá suporte a leituras e gravações. O servidor autônomo não pode se tornar uma réplica novamente.

```
az postgres server replica stop --name mydemoserver-replica --resource-group myresourcegroup
```

Excluir um servidor primário ou de réplica

Para excluir um servidor de réplica ou primário, use o comando [az postgres server delete](#).

Ao excluir um grupo de servidores primário, a replicação para todas as réplicas de leitura será interrompida. As réplicas de leitura tornam-se servidores independentes que agora têm suporte para leitura e gravação.

```
az postgres server delete --name myserver --resource-group myresourcegroup
```

API REST

Você pode criar e gerenciar réplicas de leitura usando a [API REST do Azure](#).

Preparar o servidor primário

1. Verifique o valor do servidor primário `azure.replication_support`. Ele deve ser pelo menos REPLICA para que réplicas de leitura funcionem.

```
GET  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DBforPostgreSQL/servers/{masterServerName}/configurations/azure.replication_support?  
api-version=2017-12-01
```

2. Se `azure.replication_support` não for pelo menos a REPLICA, defina-a.

```
PUT  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DBforPostgreSQL/servers/{masterServerName}/configurations/azure.replication_support?  
api-version=2017-12-01
```

```
{  
  "properties": {  
    "value": "replica"  
  }  
}
```

3. Reinicie o servidor para aplicar a alteração.

```
POST  
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DBforPostgreSQL/servers/{masterServerName}/restart?api-version=2017-12-01
```

Criar uma réplica de leitura

Você pode criar uma réplica de leitura usando [Criar API](#):

PUT

`https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DBforPostgreSQL/servers/{replicaName}?api-version=2017-12-01`

```
{  
  "location": "southeastasia",  
  "properties": {  
    "createMode": "Replica",  
    "sourceServerId":  
      "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DBforPostgreSQL/serv  
ers/{masterServerName}"  
  }  
}
```

NOTE

Para saber mais sobre em quais regiões você pode criar uma réplica, visite o artigo [conceitos de réplica de leitura](#).

Se você não tiver definido o `azure.replication_support` parâmetro para **REPLICA** em um servidor primário Otimizado para Memória ou de Uso Geral e reiniciado o servidor, você receberá uma notificação de erro. Conclua essas duas etapas antes de criar uma réplica.

Uma réplica é criada usando as mesmas configurações de computação e armazenamento que o mestre. Depois de criar uma réplica, várias configurações podem ser alteradas independentemente do servidor primário: o período de retenção de backup, o armazenamento, as vCores e a geração da computação. O tipo de preço também pode ser alterado de forma independente, exceto de ou para a camada básica.

IMPORTANT

Antes que uma configuração do grupo de servidores primário seja atualizada para um novo valor, atualize a configuração de réplica para um valor igual ou maior. Esta ação ajuda a réplica a acompanhar as alterações feitas ao mestre.

Listar réplicas

Você pode exibir a lista de réplicas de um servidor primário usando a [API da lista de réplicas](#):

GET

`https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Mic
rosoft.DBforPostgreSQL/servers/{masterServerName}/Replicas?api-version=2017-12-01`

Parar a replicação para um servidor de réplica

Você pode interromper a replicação entre um servidor primário e uma réplica de leitura usando a [API de atualização](#).

Depois de interromper a replicação para um servidor primário e uma réplica de leitura, isso não poderá ser desfeito. A réplica de leitura se torna um servidor autônomo que dá suporte a leituras e gravações. O servidor autônomo não pode se tornar uma réplica novamente.

PATCH

`https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Mic
rosoft.DBforPostgreSQL/servers/{replicaServerName}?api-version=2017-12-01`

```
{  
  "properties": {  
    "replicationRole":"None"  
  }  
}
```

Excluir um servidor primário ou de réplica

Para excluir um servidor primário ou de réplica, use a [API de exclusão](#):

Ao excluir um grupo de servidores primário, a replicação para todas as réplicas de leitura será interrompida. As réplicas de leitura tornam-se servidores independentes que agora têm suporte para leitura e gravação.

DELETE

`https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DBforPostgreSQL/servers/{serverName}?api-version=2017-12-01`

Próximas etapas

- Saiba mais sobre [réplicas de leitura no Banco de Dados do Azure para PostgreSQL](#).
- Saiba como [criar e gerenciar réplicas de leitura no portal do Azure](#).

Como criar e gerenciar réplicas de leitura no Banco de Dados do Azure para PostgreSQL usando o PowerShell

21/05/2021 • 3 minutes to read

Neste artigo, você aprenderá a criar e gerenciar réplicas de leitura no serviço Banco de Dados do Azure para PostgreSQL usando o PowerShell. Para saber mais sobre réplicas de leitura, confira [Visão Geral](#).

Azure PowerShell

Crie e gerencie réplicas de leitura usando o PowerShell.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Do módulo AZ do PowerShell instalado localmente ou do [Azure Cloud Shell](#) no navegador
- Um [servidor de Banco de Dados do Azure para PostgreSQL](#)

IMPORTANT

Enquanto o módulo Az.PostgreSQL PowerShell está em versão prévia, você precisa instalá-lo separadamente do módulo Az PowerShell usando o seguinte comando: `Install-Module -Name Az.PostgreSQL -AllowPrerelease`. Depois que o módulo Az.PostgreSQL PowerShell estiver em disponibilidade geral, ele passará a fazer parte das versões futuras do módulo do Az PowerShell e estará disponível nativamente no Azure Cloud Shell.

Ao optar por usar o PowerShell no local, conecte-se à sua conta do Azure usando o cmdlet [Connect-AzAccount](#).

Usar o Azure Cloud Shell

O Azure hospeda o Azure Cloud Shell, um ambiente de shell interativo que pode ser usado por meio do navegador. É possível usar o bash ou o PowerShell com o Cloud Shell para trabalhar com os serviços do Azure. É possível usar os comandos pré-instalados do Cloud Shell para executar o código neste artigo sem precisar instalar nada no seu ambiente local.

Para iniciar o Azure Cloud Shell:

OPÇÃO	EXEMPLO/LINK
Selecione Experimente no canto superior direito de um bloco de código. Selecionar Experimente não copia automaticamente o código para o Cloud Shell.	
Acesse https://shell.azure.com ou selecione o botão Iniciar o Cloud Shell para abri-lo no navegador.	
Selecione o botão Cloud Shell na barra de menus no canto superior direito do portal do Azure .	

Para executar o código neste artigo no Azure Cloud Shell:

1. Inicie o Cloud Shell.
2. Clique no botão **Copiar** no bloco de código para copiá-lo.
3. Cole o código na sessão do Cloud Shell ao pressionar **Ctrl+Shift+V** no Windows e no Linux ou **Cmd+Shift+V** no macOS.
4. Pressione **Enter** para executar o código.

IMPORTANT

O recurso de réplica de leitura está disponível apenas para bancos de dados do Azure para servidores PostgreSQL nas camadas de preços de uso geral ou de otimização de memória. Verifique se o servidor primário está em um desses tipos de preços.

Criar uma réplica de leitura

Um servidor de réplica de leitura pode ser criado usando o seguinte comando:

```
Get-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup |  
New-AzPostgreSqlServerReplica -Name mydemoreplicaserver -ResourceGroupName myresourcegroup
```

O comando `New-AzPostgreSqlServerReplica` exige os seguintes parâmetros:

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
ResourceGroupName	myresourcegroup	O grupo de recursos em que o servidor de réplica é criado.
Nome	mydemoreplicaserver	O nome do novo servidor de réplica criado.

Para criar uma réplica de leitura entre regiões, use o parâmetro **Localização**. O exemplo a seguir cria uma réplica de na região **Oeste dos EUA**.

```
Get-AzPostgreSqlServer -Name mrdemoserver -ResourceGroupName myresourcegroup |  
New-AzPostgreSQLServerReplica -Name mydemoreplicaserver -ResourceGroupName myresourcegroup -Location  
westus
```

Para saber mais sobre em quais regiões você pode criar uma réplica, visite o artigo [conceitos de réplica de leitura](#).

Por padrão, as réplicas de leitura são criadas com a mesma configuração de servidor que o primário, a menos que o parâmetro **SKU** seja especificado.

NOTE

É recomendável que a configuração do servidor de réplica seja mantida com valores iguais ou maiores que o primário para garantir que a réplica seja capaz de acompanhar o mestre.

Listar réplicas para um servidor primário

Para ver todas as réplicas de determinado servidor primário, execute o seguinte comando:

```
Get-AzPostgreSQLReplica -ResourceGroupName myresourcegroup -ServerName mydemoserver
```

O comando `Get-AzPostgreSQLReplica` exige os seguintes parâmetros:

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
ResourceGroupName	myresourcegroup	O grupo de recursos para o qual o servidor de réplica será criado.
ServerName	mydemoserver	O nome ou ID do servidor primário.

Parar um servidor de réplica

Parar um servidor de réplica de leitura promove a réplica de leitura como um servidor independente. Isso pode ser feito executando o cmdlet `Update-AzPostgreSqlServer` e definindo o valor de `ReplicationRole` como `None`.

```
Update-AzPostgreSqlServer -Name mydemoreplicaserver -ResourceGroupName myresourcegroup -ReplicationRole None
```

Excluir um servidor de réplica

A exclusão de um servidor de réplica de leitura pode ser feita executando o cmdlet `Remove-AzPostgreSqlServer`.

```
Remove-AzPostgreSqlServer -Name mydemoreplicaserver -ResourceGroupName myresourcegroup
```

Excluir um servidor primário

IMPORTANT

A exclusão de um servidor primário interrompe a replicação para todos os servidores primários e exclui o próprio servidor mestre. Os servidores de réplica tornam-se servidores independentes que agora suportam leitura e gravação.

Para excluir um servidor primário, você pode executar o cmdlet `Remove-AzPostgreSqlServer`.

```
Remove-AzPostgreSqlServer -Name mydemoserver -ResourceGroupName myresourcegroup
```

Próximas etapas

[Reiniciar o Banco de Dados do Azure para PostgreSQL usando o PowerShell](#)

Mover um servidor único do Banco de Dados do Azure para PostgreSQL para outra região usando o portal do Azure

21/05/2021 • 3 minutes to read

Há vários cenários para mover um servidor existente do Banco de Dados do Azure para PostgreSQL de uma região para outra. Por exemplo, talvez você queira mover um servidor de produção para outra região como parte de seu planejamento de recuperação de desastre.

Você pode usar uma [réplica de leitura entre regiões](#) do Banco de Dados do Azure para PostgreSQL para concluir a mudança para outra região. Para fazer isso, primeiro crie uma réplica de leitura na região de destino. Em seguida, interrompa a replicação para o servidor de réplica de leitura para torná-lo um servidor autônomo que aceita tráfego de leitura e gravação.

NOTE

Este artigo se concentra em mover seu servidor para uma região diferente. Se você quiser mover o servidor para um grupo de recursos ou assinatura diferente, consulte o artigo [mover](#).

Pré-requisitos

- O recurso de réplica de leitura entre regiões está disponível apenas para Banco de Dados do Azure para PostgreSQL - Servidor Único nos tipos de preços de Uso Geral ou Otimizado para Memória. Verifique se o servidor de origem está em um desses tipos de preços.
- Verifique se o servidor de origem do Banco de Dados do Azure para PostgreSQL está na região do Azure da qual você deseja mover.

Preparar para mover

Para preparar o servidor de origem para replicação usando o portal do Azure, use as seguintes etapas:

1. Faça logon no [Portal do Azure](#).
2. Selecione o servidor existente do Banco de Dados do Azure para PostgreSQL que você deseja usar como servidor de origem. Essa ação abre a página **Visão geral** do runbook.
3. No menu do servidor, selecione **Replicação**. Se o suporte à replicação do Azure for definido como no mínimo **Réplica**, você poderá criar réplicas de leitura.
4. Se o suporte à replicação do Azure não estiver definido no mínimo **Réplica**, defina-o. Selecione **Salvar**.
5. Reinicie o servidor para aplicar a alteração selecionando **Sim**.
6. Você receberá duas notificações do portal do Azure quando a operação for concluída. Há uma notificação para atualizar o parâmetro do servidor. Logo em seguida, há outra notificação para a reinicialização do servidor.
7. Atualize a página do portal do Azure para atualizar a barra de ferramentas de Replicação. Agora você pode criar réplicas de leitura para esse servidor.

Para criar um servidor de réplica de leitura entre regiões na região de destino usando o portal do Azure, execute as seguintes etapas:

1. Selecione o servidor existente do Banco de Dados do Azure para PostgreSQL que você deseja usar como servidor de origem.
2. Selecione **Replicação** no menu, em **CONFIGURAÇÕES**.
3. Selecione **para adicionar réplica**.
4. Insira um nome para o servidor de réplica.
5. Selecione o local para o servidor de réplica. O local padrão é o mesmo que o do servidor primário. Verifique se você selecionou o local de destino onde deseja que a réplica seja implantada.
6. Selecione **OK** para confirmar a criação da réplica. Durante a criação de réplica, os dados são copiados do servidor de origem para a réplica. O tempo de criação pode durar vários minutos ou mais, em proporção ao tamanho do servidor de origem.

NOTE

Quando você cria uma réplica, ela não herda as regras de firewall nem o ponto de extremidade de serviço da VNet do servidor primário. Essas regras precisam ser configuradas independentemente da réplica.

Mover

IMPORTANT

O servidor autônomo não pode se tornar uma réplica novamente. Antes de interromper a replicação em uma réplica de leitura, verifique se a réplica tem todos os dados de que você precisa.

Para interromper a replicação para a réplica no portal do Azure, use as seguintes etapas:

1. Depois que a réplica tiver sido criada, localize e selecione seu servidor de origem do Banco de Dados do Azure para PostgreSQL.
2. Selecione **Replicação** no menu, em **CONFIGURAÇÕES**.
3. Selecione o servidor de réplica.
4. Selecione **Parar replicação**.
5. Confirme que você deseja interromper a replicação clicando em **OK**.

Limpar o servidor de origem

Talvez você queira excluir o servidor de origem do Banco de Dados do Azure para PostgreSQL. Para fazer isso, execute as seguintes etapas:

1. Depois que a réplica tiver sido criada, localize e selecione seu servidor de origem do Banco de Dados do Azure para PostgreSQL.
2. Na janela **Visão Geral**, selecione **Excluir**.
3. Digite o nome do servidor de origem para confirmar que deseja excluir.
4. Selecione **Excluir**.

Próximas etapas

Neste tutorial, você moveu um servidor do Banco de Dados do Azure para PostgreSQL de uma região para outra usando o portal do Azure e depois limpou os recursos de origem desnecessários.

- Saiba mais sobre [ler réplicas](#)
- Saiba mais sobre [gerenciar réplica de leitura no portal do Azure](#)
- Saiba mais sobre as opções de [continuidade dos negócios](#)

Visão geral – Banco de Dados do Azure para PostgreSQL – servidor flexível

21/05/2021 • 8 minutes to read

O [Banco de Dados do Azure para PostgreSQL](#) desenvolvido com o PostgreSQL community edition está disponível em três modos de implantação:

- [Servidor Único](#)
- Servidor Flexível (versão prévia)
- Hiperescala (Citus)

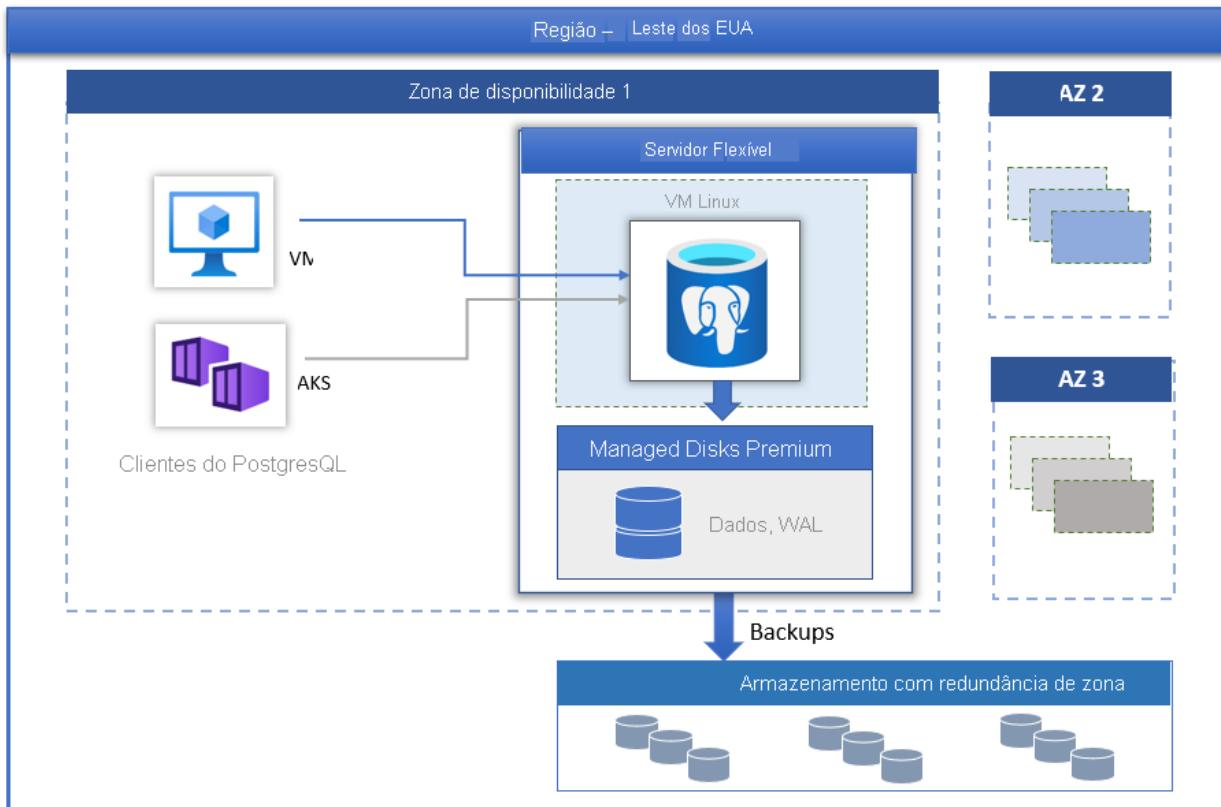
Neste artigo, forneceremos uma visão geral e uma introdução aos principais conceitos do modelo de implantação de servidor flexível.

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Visão geral

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL é um serviço de banco de dados totalmente gerenciado, projetado para proporcionar um controle mais granular e flexibilidade nas funções de gerenciamento de banco de dados e definições de configuração. Em geral, o serviço oferece mais flexibilidade e personalizações de configuração do servidor com base nos requisitos do usuário. A arquitetura de servidor flexível permite aos usuários colocar o mecanismo de banco de dados com a camada de cliente para uma latência mais baixa, escolher a alta disponibilidade em uma só zona de disponibilidade e em várias zonas de disponibilidade. Os servidores flexíveis também oferecem melhores controles de otimização de custos com a capacidade de parar/iniciar o servidor e a camada de computação expansível, que é ideal para cargas de trabalho que não precisam da capacidade de computação completa continuamente. Atualmente, o serviço dá suporte à versão da comunidade do PostgreSQL 11 e 12. No momento, o serviço está em versão prévia, disponível hoje mesmo em uma ampla variedade de [regiões do Azure](#).



Os servidores flexíveis são mais adequados para

- Desenvolvimentos de aplicativos que exigem controle e personalizações melhores.
- Alta disponibilidade com redundância de zona
- Janelas de manutenção gerenciadas

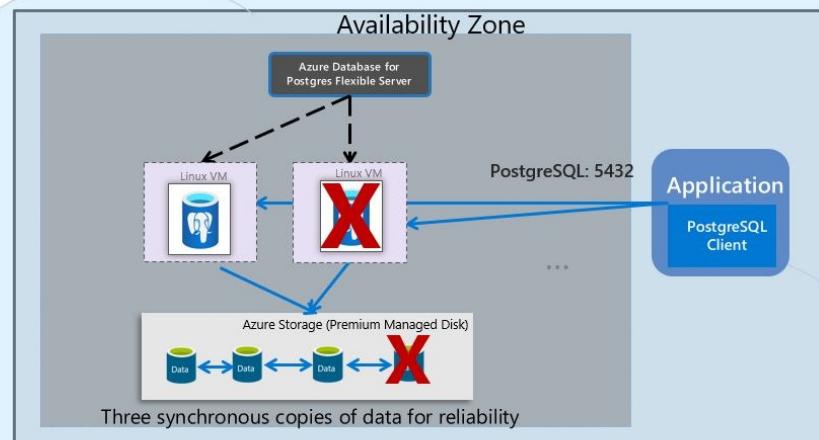
Alta disponibilidade

O modelo de implantação de servidor flexível foi projetado para dar suporte à alta disponibilidade em uma só zona de disponibilidade e em várias zonas de disponibilidade. A arquitetura separa a computação do armazenamento. O mecanismo de banco de dados é executado em uma máquina virtual do Linux, enquanto os arquivos residem no Armazenamento do Azure. O armazenamento mantém três cópias síncronas localmente redundantes dos arquivos de banco de dados, garantindo a durabilidade dos dados.

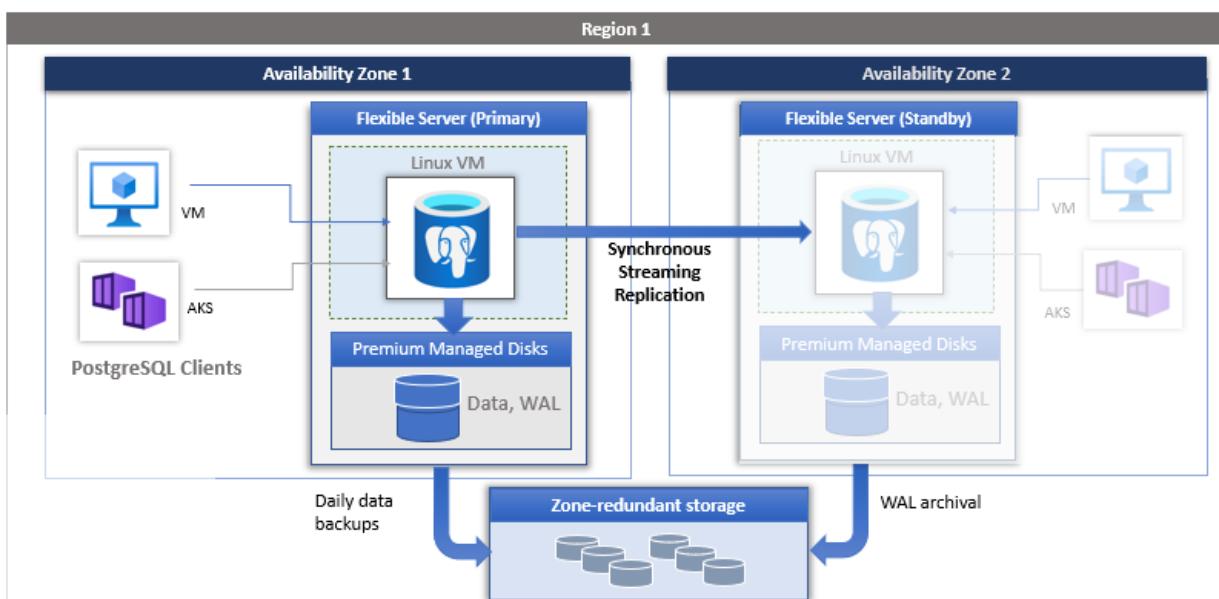
Durante eventos de failover planejados ou não planejados, se o servidor ficar inativo, o serviço manterá a alta disponibilidade dos servidores usando o seguinte procedimento automatizado:

1. Uma nova VM de computação do Linux será provisionada.
2. O armazenamento com os arquivos de dados é mapeado para a nova Máquina Virtual
3. O mecanismo de banco de dados PostgreSQL é colocado online na nova Máquina Virtual.

A imagem abaixo mostra a transição para a falha de VM e de armazenamento.



Se a alta disponibilidade com redundância de zona estiver configurada, o serviço provisionará e manterá um servidor em espera ativa na zona de disponibilidade, na mesma região do Azure. As alterações de dados no servidor de origem são replicadas de maneira síncrona para o servidor em espera para garantir zero perda de dados. Com alta disponibilidade com redundância de zona, uma vez que o evento de failover planejado ou não planejado é disparado, o servidor em espera fica online imediatamente e está disponível para processar transações de entrada. Isso permite a resiliência de serviço de uma falha de zona de disponibilidade em uma região do Azure que dá suporte a várias zonas de disponibilidade, conforme mostrado na imagem abaixo.



Confira o [documento sobre alta disponibilidade](#) para obter mais detalhes.

Aplicação de patch automatizada com janela de manutenção gerenciada

O serviço executa a aplicação automatizada de patch do hardware, do sistema operacional e do mecanismo de banco de dados subjacentes. A aplicação de patch inclui atualizações de segurança e software. Para o mecanismo PostgreSQL, as atualizações de versão secundárias também são incluídas como parte da versão de manutenção planejada. Os usuários podem configurar o agendamento de aplicação de patch para que ele seja gerenciado pelo sistema ou definir um agendamento personalizado. Durante o agendamento de manutenção, o patch é aplicado e o servidor pode precisar ser reiniciado como parte do processo de aplicação de patch para

concluir a atualização. Com o agendamento personalizado, os usuários podem tornar o ciclo de aplicação de patch previsível e escolher uma janela de manutenção com impacto mínimo sobre os negócios. No geral, o serviço segue a agenda de lançamento mensal como parte do lançamento e da integração contínua.

Backups automáticos

O serviço de servidor flexível cria automaticamente backups de servidor e os armazena no usuário configurado no local em ZRS (com redundância de zona). Os backups podem ser usados para restaurar o servidor em qualquer ponto no tempo dentro do período de retenção de backup. O período de retenção de backup padrão é de sete dias. A retenção pode ser configurada opcionalmente em até 35 dias. Todos os backups são criptografados usando a criptografia AES de 256 bits. Confira [Backups](#) para obter mais detalhes.

Ajustar o desempenho e a escala em segundos

O serviço de servidor flexível está disponível em três camadas de computação: Expansível, Uso Geral e Otimizado para Memória. A camada Expansível é mais adequada para desenvolvimento de baixo custo e cargas de trabalho de simultaneidade baixa que não precisam da capacidade de computação completa continuamente. O Uso Geral e o Otimizado para Memória são mais adequados para cargas de trabalho de produção que exigem alta simultaneidade, escala e desempenho previsível. Você pode criar seu primeiro aplicativo em um banco de dados pequeno por alguns dólares por mês e, depois, ajustar a escala de acordo com as necessidades da sua solução.

Parar/iniciar o servidor para reduzir o TCO

O serviço de servidor flexível permite que você pare e inicie o servidor sob demanda para reduzir o TCO. A cobrança da camada de computação é interrompida imediatamente quando o servidor é parado. Com isso, você pode ter uma redução de custo significativa durante o desenvolvimento e o teste e para cargas de trabalho de produção previsíveis com limite de tempo. O servidor permanece no estado parado por sete dias, a menos que ele seja reiniciado antes.

Segurança de nível empresarial

O serviço de servidor flexível usa o módulo de criptografia validado por FIPS 140-2 para a criptografia de armazenamento de dados em repouso. Os dados, incluindo backups, e os arquivos temporários criados durante a execução de consultas são criptografados. O serviço usa a criptografia AES de 256 bits incluída na criptografia de armazenamento do Azure, e as chaves podem ser gerenciadas pelo sistema (padrão). O serviço criptografa os dados em movimento com o protocolo SSL/TLS imposto por padrão. O serviço só impõe a versão do TLS 1.2 e só dá suporte a ela.

Os servidores flexíveis permitem acesso privado completo aos servidores usando a rede virtual do Azure (Integração VNET). Os servidores da rede virtual do Azure só podem ser acessados e conectados por meio de endereços IP privados. Com a Integração VNET, o acesso público é negado e os servidores não podem ser acessados por meio de pontos de extremidade públicos.

Monitoramento e alertas

O serviço de servidor flexível é equipado com recursos internos de monitoramento e alerta de desempenho. Todas as métricas do Azure têm uma frequência de um minuto e cada uma delas fornece 30 dias de histórico. É possível configurar alertas nas métricas. O serviço expõe as métricas do servidor host para monitorar a utilização de recursos e permite configurar logs de consultas lentas. Usando essas ferramentas, você pode otimizar rapidamente suas cargas de trabalho e configurar seu servidor para ter o melhor desempenho.

PgBouncer interno

O servidor flexível vem com um PgBouncer interno, um pooler de conexão. Opcionalmente, você pode habilitá-lo e conectar seus aplicativos ao servidor de banco de dados por meio do PgBouncer usando o mesmo nome do host e a porta 6432.

Regiões do Azure

Uma das vantagens de executar sua carga de trabalho no Azure é obter um alcance global. O servidor flexível está disponível hoje nas seguintes regiões do Azure:

REGIÃO	DISPONIBILIDADE	HA COM REDUNDÂNCIA DE ZONA
Europa Ocidental	✓	✓
Norte da Europa	✓	✓
Sul do Reino Unido	✓	✓
Leste dos EUA 2	✓	✓
Oeste dos EUA 2	✓	✓
Centro dos EUA	✓	✓
Leste dos EUA	✓	✓
Sudeste Asiático	✓	✓
Japan East	✓	✓
Leste da Austrália	✓	✓
Canadá Central	✓	✓
França Central	✓	✓

Continuamos adicionando mais regiões para o servidor flexível.

Migração

O serviço executa a versão da comunidade do PostgreSQL. Isso permite a compatibilidade total do aplicativo e exige o mínimo de custo de refatoração para migrar um aplicativo existente desenvolvido no mecanismo PostgreSQL para um Servidor Flexível.

- **Despejo e restauração:** nas migrações offline, em que os usuários podem ter algum tempo de inatividade, realizar o despejo e a restauração com ferramentas da comunidade, como pg_dump e pg_restore, pode fornecer uma forma mais rápida de migração. Confira [Migrar usando despejo e restauração](#) para obter detalhes.
- **Serviço de Migração de Banco de Dados do Azure:** para migrações diretas e simplificadas para um servidor flexível com tempo de inatividade mínimo, você pode aproveitar o Serviço de Migração de Banco de Dados do Azure. Confira [DMS por meio do portal](#) e [DMS por meio da CLI](#). Você pode fazer a migração por meio do Banco de Dados do Azure para PostgreSQL – Servidor Único para Servidor Flexível. Confira este [artigo sobre o DMS](#) para obter detalhes.

Contatos

Para perguntas ou sugestões sobre o servidor flexível do Banco de Dados do Azure para PostgreSQL, envie um email para a equipe do Banco de Dados do Azure para PostgreSQL ([@Ask BD do Azure para PostgreSQL](#)). Observe que esse endereço de email não é um alias de suporte técnico.

Além disso, considere os seguintes pontos de contato, conforme apropriado:

- Para entrar em contato com o Suporte do Azure, [crie um tíquete no Portal do Azure](#).
- Para corrigir um problema com sua conta, apresente uma [solicitação de suporte](#) no portal do Azure.
- Para fornecer comentários ou solicitar novos recursos, crie uma entrada por meio do [UserVoice](#).

Próximas etapas

Agora que você leu uma introdução ao modo de implantação de servidor flexível do Banco de Dados do Azure para PostgreSQL, você está pronto para criar seu primeiro servidor: [Criar um Servidor Flexível do Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#)

Notas de versão – Banco de Dados do Azure para PostgreSQL – Servidor Flexível

15/07/2021 • 2 minutes to read

Esta página fornece notícias e atualizações mais recentes sobre adições de recursos, suporte para versões de mecanismos, extensões e qualquer outro anúncio relevante para o Servidor Flexível – PostgreSQL.

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Versão: 25 de maio de 2021

- Suporte à [versão principal 13 do PostgreSQL](#).
- Suporte a extensões, incluindo pg_partman, pg_cron e pgaudit. Confira a [página de extensões](#) para obter as versões com suporte em cada versão principal.
- Várias correções de bug e melhorias de desempenho e estabilidade.

Versão: 26 de abril de 2021

- Suporte para os [secundários mais recentes do PostgreSQL](#) 12.6 e 11.11 com a criação de servidores.
- Suporte para a [zona DNS privada](#) da VNET (rede virtual).
- Suporte para escolher a zona de disponibilidade durante a operação de recuperação pontual.
- Suporte para novas [regiões](#), incluindo o Leste da Austrália, o Canadá Central e a França Central.
- Suporte para o pooler de conexão [PgBouncer interno](#).
- [Desempenho inteligente](#) em visualização pública.
- Várias correções de bug e melhorias de desempenho e estabilidade.

Contatos

Para perguntas ou sugestões sobre o servidor flexível do Banco de Dados do Azure para PostgreSQL, envie um email para a equipe do Banco de Dados do Azure para PostgreSQL ([@Ask BD do Azure para PostgreSQL](#)). Observe que esse endereço de email não é um alias de suporte técnico.

Além disso, considere os seguintes pontos de contato, conforme apropriado:

- Para entrar em contato com o Suporte do Azure, [crie um tiquete no Portal do Azure](#).
- Para corrigir um problema com sua conta, apresente uma [solicitação de suporte](#) no portal do Azure.
- Para fornecer comentários ou solicitar novos recursos, [crie uma entrada por meio do UserVoice](#).

Próximas etapas

Agora que você leu uma introdução ao modo de implantação de servidor flexível do Banco de Dados do Azure para PostgreSQL, você está pronto para criar seu primeiro servidor: [Criar um Servidor Flexível do Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#)

Início Rápido: Criar um Servidor Flexível do Banco de Dados do Azure para PostgreSQL no portal do Azure

15/07/2021 • 8 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Banco de Dados do Azure para PostgreSQL é um serviço gerenciado usado para executar, gerenciar e dimensionar bancos de dados altamente disponíveis do PostgreSQL na nuvem. Este guia de início rápido mostra como criar um Servidor Flexível do Banco de Dados do Azure para PostgreSQL usando o portal do Azure em aproximadamente cinco minutos.

Caso você não tenha uma assinatura do Azure, crie uma [conta gratuita do Azure](#) antes de começar.

Entre no Portal do Azure

Abra o navegador da Web e acesse o [portal](#). Insira suas credenciais para entrar no portal. A exibição padrão é o painel de serviço.

Criar um Banco de Dados do Azure para o servidor PostgreSQL

Um Banco de Dados do Azure para PostgreSQL é criado com um conjunto configurado de [recursos de computação e armazenamento](#). O servidor é criado dentro de um [Grupo de recursos do Azure](#).

Para criar um Banco de Dados do Azure para o servidor PostgreSQL, execute as seguintes etapas:

1. Selecione **Criar um recurso** (+) no canto superior esquerdo do portal.
2. Selecione **Bancos de Dados > Banco de Dados do Azure para PostgreSQL**.

The screenshot shows the Azure Marketplace 'New' page. In the top left, there's a breadcrumb navigation: Home > New. Below it is a search bar with the placeholder 'Search the Marketplace'. The main area has two tabs at the top: 'Azure Marketplace' and 'See all'. Underneath are two sections: 'Featured' and 'See all'. The 'Featured' section contains several items with icons and names: 'Get started' (cloud icon), 'Recently created' (recent document icon), 'AI + Machine Learning' (AI icon), 'Analytics' (bar chart icon), 'Blockchain' (blockchain icon), 'Compute' (server icon), 'Containers' (cubes icon), 'Databases' (database icon, which is highlighted with a light gray background), 'Developer Tools' (code editor icon), 'DevOps' (gears icon), 'Identity' (key icon), 'Integration' (integration icon), and 'Internet of Things' (IoT icon). The 'Databases' item has a dashed blue rectangular box around it. To the right of the 'Databases' section, there are three more items: 'Azure SQL Managed Instance' (cloud icon), 'SQL Database' (SQL icon), and 'Azure Synapse Analytics (formerly SQL DW)' (hexagon icon). Each item has a 'Quickstarts + tutorials' link below its name.

3. Selecione a opção de implantação Servidor flexível.

Home >

Select Azure Database for PostgreSQL deployment option

Microsoft

How do you plan to use the service?

Single server
Best for broad range of traditional transactional workloads.

Enterprise ready, fully managed community PostgreSQL server with up to 64 vCores, optional geospatial support, full-text search and more.

[Create](#) [Learn more](#)

Flexible server (Preview)
Next generation Azure Database for PostgreSQL offering in preview.

Preview is not recommended for production workloads. Certain features might not be supported or have constrained capabilities.

[Create](#) [Learn more](#)

Hyperscale (Citus) server group
Best for ultra-high performance and data needs beyond 100GB.

Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response. Supports both transactional/operational workloads and hybrid transactional analytics workloads.

[Create](#) [Learn more](#)

4. Preencha o formulário Básico com as seguintes informações:

Detalhes do servidor

Insira as configurações necessárias para o servidor, incluindo a escolha de um local e a configuração dos recursos.

Nome do servidor*

mydemoserver-pg



Região * ⓘ

Leste dos EUA



Produção (tamanho pequeno)

(medio)



Produção (tamanho grande)



Desenvolvimento



Adequado para produção, pré-produção ou teste de bancos de dados de pequeno a médio porte com requisitos de simultaneidade de baixo a médio.

Computação e armazenamento

Uso Geral, D2s_v3

2 vCores, 8 GiB RAM, 128 GiB de armazenamento

[Configurar servidor](#)

Zona de disponibilidade

1



HA com redundância de zona

Versão do PostgreSQL*

13



Conta de administrador

Nome de usuário do administrador *

meuadmin



Senha * ⓘ



Confirmar senha *



CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Subscription	O nome da sua assinatura	A assinatura do Azure que você deseja usar para o servidor. Caso você tenha várias assinaturas, escolha a assinatura na qual você deseja receber a cobrança do recurso.

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Resource group	<i>myresourcegroup</i>	Um novo nome do grupo de recursos ou um existente de sua assinatura.
Tipo de carga de trabalho	Seleção de SKU padrão	Você pode escolher entre desenvolvimento (SKU com capacidade de intermitência), produção pequena/média (SKU de uso geral) ou produção grande (SKU com otimização de memória). Você pode personalizar ainda mais a SKU e o armazenamento clicando em <i>Configurar link do servidor</i> .
Zona de disponibilidade	Seu AZ preferido	Você pode escolher em qual zona de disponibilidade deseja que o servidor seja implantado. Isso é útil para colocar o servidor ao lado do seu aplicativo. Se você escolher <i>Nenhuma preferência</i> , um AZ padrão será selecionado para você.
Alta disponibilidade	Habilitar a implantação com redundância de zona	Ao selecionar essa opção, um servidor em espera com a mesma configuração que o primário será provisionado automaticamente em uma zona de disponibilidade diferente na mesma região. Observação: você também pode habilitar ou desabilitar a criação de alta disponibilidade do servidor de postagem.
Nome do servidor	<i>mydemoserver-pg</i>	Um nome exclusivo que identifica o Banco de Dados do Azure para o servidor PostgreSQL. O nome de domínio <i>postgres.database.azure.com</i> é acrescentado ao nome do servidor fornecido. O servidor pode conter apenas letras minúsculas, números e o caractere de hífen (-). Ele deve conter, pelo menos, 3 até 63 caracteres.
Nome de usuário do administrador	<i>myadmin</i>	Sua própria conta de logon para uso ao se conectar ao servidor. O nome de logon do administrador não pode ser azure_superuser , azure_pg_admin , admin , administrator , root , guest ou public . Ele não pode começar com pg_ .

CONFIGURAÇÃO	VALOR SUGERIDO	DESCRIÇÃO
Senha	Sua senha	Uma nova senha para a conta do administrador do servidor. Ele deve conter entre 8 e 128 caracteres. A senha precisa conter caracteres de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números (0 a 9) e caracteres não alfanuméricos (!, \$, #, % etc.).
Location	A região mais próxima dos usuários	A localização mais próxima dos usuários.
Versão	A última versão principal	A última versão principal do PostgreSQL, a menos que você tenha requisitos específicos.
Computação + armazenamento	Uso Geral, 4 vCores, 512 GB, 7 dias	As configurações de computação, armazenamento e backup para o novo servidor. Selecione Configurar servidor . <i>Uso Geral, 4 vCores, 512 GB, and 7 dias</i> são os valores padrão de Camada de computação , vCore , Armazenamento e Período de Retenção de Backup . Você pode deixar esses controles deslizantes como estão ou ajustá-los. Para salvar a seleção desse tipo de preço, selecione OK . A captura de tela a seguir demonstra essas seleções.

Compute + storage

Compute resources are pre-allocated and billed per minute based on vCores configured.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (4-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (4-64 vCores) - Best for workloads that require a high memory to CPU ratio

vCore



Storage size (in GiB)



UP TO **5000** AVAILABLE IOPS

Backups

Configure automatic server backups that can be used to restore your server to a point-in-time.

Retention period



Na guia Rede, você pode escolher como o servidor poderá ser acessado. O Banco de Dados do Azure para PostgreSQL cria um firewall no nível do servidor. Ele impede que os aplicativos e as ferramentas externas se conectem ao servidor e aos bancos de dados no servidor, a menos que uma regra seja criada para abrir o firewall em endereços IP específicos. Recomendamos tornar o servidor publicamente acessível:

E, depois, restringi-lo ao seu endereço IP do cliente:

Se você escolher acesso privado

6. Selecione **Revisar + criar** para revisar suas seleções. Selecione **Criar** para provisionar o servidor. Esta operação pode levar alguns minutos.
7. Na barra de ferramentas, selecione o ícone (sino) **Notificações** para monitorar o processo de implantação. Depois que a implantação é feita, você pode selecionar **Fixar no painel**, que cria um bloco para esse servidor no seu painel do portal do Azure como um atalho para a página **Visão geral** do servidor. A opção **Ir para recurso** abre a página **Visão geral** do servidor.

Por padrão, um banco de dados **postgres** é criado no servidor. O banco de dados **postgres** é um banco

de dados padrão destinado ao uso dos usuários, de utilitários e de aplicativos de terceiros. (O outro banco de dados padrão é o **azure_maintenance**. Sua função é separar os processos de serviço gerenciado das ações do usuário. Não é possível acessar este banco de dados).

NOTE

As conexões ao Banco de Dados do Azure para servidor PostgreSQL se comunicam pela porta 5432. Quando você tenta se conectar em uma rede corporativa, talvez o tráfego de saída pela porta 5432 não seja permitido pelo firewall da rede. Se isso acontecer, você não conseguirá se conectar ao servidor, a menos que o departamento de TI abra a porta 5432.

Obter informações de conexão

Ao criar o Banco de Dados do Azure para o servidor PostgreSQL, um banco de dados padrão chamado **postgres** é criado. Para se conectar ao servidor de banco de dados, você precisa do nome do servidor completo e das credenciais de logon do administrador. Talvez você tenha anotado esses valores anteriormente no artigo do Guia de início rápido. Caso não tenha anotado, poderá encontrar facilmente o nome do servidor e as informações de logon na página **Visão geral** do servidor no portal.

Abra a página **Visão geral** do servidor. Anote o **Nome do servidor** e o **Nome de logon do administrador do servidor**. Focalize o cursor em cada campo e o símbolo de cópia será exibido à direita do texto. Selecione o símbolo de cópia, conforme necessário, para copiar os valores.

Conectar-se ao banco de dados PostgreSQL usando psql

Há vários aplicativos que você pode usar para conectar o servidor Banco de Dados do Azure para PostgreSQL. Se o computador cliente tiver o PostgreSQL instalado, você poderá usar uma instância local do **psql** para se conectar a um servidor PostgreSQL do Azure. Usaremos agora o utilitário de linha de comando **psql** para nos conectarmos ao servidor PostgreSQL do Azure.

1. Execute o comando **psql** a seguir para se conectar a um Banco de Dados do Azure para servidor PostgreSQL

```
psql --host=<servername> --port=<port> --username=<user> --dbname=<dbname>
```

Por exemplo, o comando a seguir se conecta ao banco de dados padrão chamado **postgres** no seu servidor PostgreSQL **mydemoserver.postgres.database.azure.com** usando as credenciais de acesso. Insira o **<server_admin_password>** que você escolheu quando uma senha foi solicitada a você.

```
psql --host=mydemoserver-pg.postgres.database.azure.com --port=5432 --username=myadmin --dbname=postgres
```

Depois que você se conectar, o utilitário **psql** exibirá um prompt do **postgres** no qual você digitará os comandos do sql. Na saída de conexão inicial, um aviso pode ser exibido, pois o **psql** que você está

usando pode ter uma versão diferente da versão Banco de Dados do Azure para o servidor PostgreSQL.

Exemplo de saída psql:

```
psql (12.3 (Ubuntu 12.3-1.pgdg18.04+1), server 13.2)
WARNING: psql major version 12, server major version 13.
        Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=>
```

TIP

Se o firewall não está configurado para permitir o endereço IP do seu cliente, ocorre o seguinte erro:

"psql: FATAL: nenhuma entrada pg_hba.conf para o host <IP address>, usuário "myadmin", banco de dados "postgres", SSL em FATAL: Uma conexão SSL é necessária. Especifique as opções de SSL e tente novamente.

Confirme se o que IP do seu cliente é permitido na etapa de regras de firewall acima.

2. Crie um banco de dados chamado "mypgsqldb" em branco no prompt digitando o seguinte comando:

```
CREATE DATABASE mypgsqldb;
```

3. No prompt, execute o seguinte comando para mudar as conexões para o banco de dados **mypgsqldb** recém-criado:

```
\c mypgsqldb
```

4. Digite **\q** e selecione a tecla Enter para sair do psql.

Você está conectado ao Banco de Dados do Azure para servidor PostgreSQL via psql e criou um banco de dados do usuário em branco.

Limpar os recursos

Limpe os recursos criados no Guia de início rápido usando uma das duas maneiras. Você pode excluir o grupo de recursos do Azure, que inclui todos os recursos no grupo de recursos. Se desejar manter os outros recursos intactos, exclua apenas o recurso de servidor.

TIP

Outros Guias de Início Rápido na coleção aproveitam este Guia de Início Rápido. Se você pretende continuar trabalhando com Guias de início rápido, não limpe os recursos criados neste Guia de início rápido. Caso contrário, siga estas etapas para excluir os recursos que foram criados por este Guia de início rápido no portal.

Para excluir o grupo de recursos inteiro, incluindo o servidor recém-criado:

1. Localize o grupo de recursos no portal. No menu à esquerda, selecione **Grupos de recursos**. Em seguida, selecione o nome do grupo de recursos, como o exemplo, **myresourcegroup**.
2. Na página do grupo de recursos, selecione **Excluir**. Insira o nome do grupo de recursos, como o exemplo **myresourcegroup**, na caixa de texto para confirmar a exclusão. Selecione **Excluir**.

Para excluir apenas o servidor recém-criado:

1. Localize o servidor no portal, caso você não esteja com ele aberto. No menu à esquerda, selecione **Todos os recursos**. Em seguida, procure o servidor que você criou.
2. Na página **Visão Geral**, selecione **Excluir**.



The screenshot shows the Azure portal interface for managing a PostgreSQL server. The top navigation bar includes 'mydemoserver-pg', a search bar, and various action buttons like 'Excluir' (Delete), 'Redefinir senha' (Reset password), 'Restaurar' (Restore), and 'Relançar' (Relaunch). Below the navigation, there's a sidebar with links such as 'Visão geral', 'Log de atividades', 'Controle de acesso (IAM)', 'Marcas', 'Configurações', 'Computação e armazenamento', and 'Rede'. The main content area displays general information about the server, including its name ('mydemoserver-pg'), resource group ('myresourcegroup'), status ('Disponível'), location ('Leste dos EUA'), subscription ('My Subscription'), and other details like IP configuration and RAM. The 'Excluir' button is prominently highlighted with a red box.

3. Confirme o nome do servidor que deseja excluir e veja embaixo do nome dele os bancos de dados que são afetados. Insira o nome do servidor na caixa de texto, como o exemplo **mydemoserver**. Selecione **Excluir**.

Próximas etapas

[Implantar um aplicativo Django usando o Serviço de Aplicativo e o PostgreSQL](#)

Início Rápido: Criar um Servidor Flexível do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure

15/07/2021 • 4 minutes to read

Este início rápido mostrará como usar os comandos da [CLI do Azure](#) no [Azure Cloud Shell](#) para criar um Servidor Flexível do Banco de Dados do Azure para PostgreSQL em cinco minutos. Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está atualmente na versão prévia.

Iniciar o Azure Cloud Shell

O [Azure Cloud Shell](#) é um shell gratuito e interativo que poderá ser usado para executar as etapas deste artigo. Ele tem ferramentas do Azure instaladas e configuradas para usar com sua conta.

Para abrir o Cloud Shell, basta selecionar **Experimentar** no canto superior direito de um bloco de código. Você também pode abrir o Cloud Shell em uma guia separada do navegador indo até <https://shell.azure.com/bash>. Selecione **Copiar** para copiar os blocos de código, cole-o no Cloud Shell e selecione **Enter** para executá-lo.

Caso prefira instalar e usar a CLI localmente, este início rápido exigirá a CLI do Azure versão 2.0 ou posterior. Execute `az --version` para encontrar a versão. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Pré-requisitos

Você precisará fazer logon em sua conta usando o comando `az login`. Observe a propriedade **id**, que se refere à **ID da Assinatura** para sua conta do Azure.

```
az login
```

Selecione a assinatura específica em sua conta usando o comando `az account set`. Anote o valor de **id** da saída `az login` para usar como valor para o argumento **subscription** no comando. Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Para obter todas as suas assinaturas, use `az account list`.

```
az account set --subscription <subscription id>
```

Criar um servidor flexível

Crie um [grupo de recursos do Azure](#) usando o comando `az group create` e depois crie um servidor flexível do PostgreSQL dentro desse grupo de recursos. Você deve fornecer um nome exclusivo. O exemplo a seguir cria um grupo de recursos chamado `myresourcegroup` na localização `westus`.

```
az group create --name myresourcegroup --location westus
```

Crie um servidor flexível usando o comando `az postgres flexible-server create`. Um servidor pode conter vários bancos de dados. O seguinte comando criará um servidor usando valores e padrões de serviço do [contexto local](#) da CLI do Azure:

```
az postgres flexible-server create
```

O servidor criado terá os atributos abaixo:

- Nome do servidor gerado automaticamente, nome de usuário do administrador, senha de administrador, nome do grupo de recursos (caso ainda não tenha sido especificado no contexto local). Além disso, o servidor e o grupo de recursos estarão no mesmo local
- Padrões de serviço para as configurações de servidor restantes: nível de computação (Uso Geral), tamanho/SKU da computação (D2s_v3 – 2 vCore e 8 GB de RAM), período de retenção de backup (7 dias) e versão do PostgreSQL (12)
- O método de conectividade padrão será o acesso Privado (*Integração VNET*) com rede e sub-rede virtuais geradas automaticamente

NOTE

O método de conectividade não poderá ser alterado após a criação do servidor. Por exemplo, caso tenha selecionado o *acesso Privado* (*Integração VNET*) durante a criação, não será possível alterar para o *acesso Público* (*endereços IP permitidos*) posteriormente. Recomendamos criar um servidor com acesso Privado para que seja possível acessar seu servidor com segurança usando a *Integração VNet*. Saiba mais sobre o acesso Privado no [artigo sobre conceitos](#).

Caso queira alterar os padrões, consulte a documentação de referência da CLI do Azure para obter uma lista completa de parâmetros configuráveis da CLI.

NOTE

As conexões ao Banco de Dados do Azure para PostgreSQL se comunicam pela porta 5432. Se estiver tentando se conectar em uma rede corporativa, talvez o tráfego de saída pela porta 5432 não seja permitido. Nesse caso, não será possível se conectar ao servidor, a menos que o departamento de TI abra a porta 5432.

Obter informações de conexão

Para se conectar ao servidor, é preciso fornecer credenciais de acesso e informações do host.

```
az postgres flexible-server show --resource-group myresourcegroup --name mydemoserver
```

O resultado está no formato JSON. Anote o **fullyQualifiedDomainName** e o **administratorLogin**.

```
{  
    "administratorLogin": "myadmin",  
    "earliestRestoreDate": null,  
    "fullyQualifiedDomainName": "mydemoserver.postgres.database.azure.com",  
    "id": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/myresourcegroup/providers/Microsoft.DBforPostgreSQL/flexibleServers/mydemoserver  
",  
    "location": "westus",  
    "name": "mydemoserver",  
    "resourceGroup": "myresourcegroup",  
    "sku": {  
        "capacity": 2,  
        "name": "Standard_D2s_v3",  
        "size": null,  
        "tier": "GeneralPurpose"  
    },  
    "publicAccess": "Enabled",  
    "storageProfile": {  
        "backupRetentionDays": 7,  
        "geoRedundantBackup": "Disabled",  
        "storageMb": 131072  
    },  
    "tags": null,  
    "type": "Microsoft.DBforPostgreSQL/flexibleServers",  
    "userVisibleState": "Ready",  
    "version": "12"  
}
```

Conectar-se usando um cliente de linha de comando do PostgreSQL

Como o servidor flexível foi criado com *acesso privado (Integração VNet)*, será necessário se conectar ao servidor de um recurso na mesma VNet do servidor. Será possível criar uma máquina virtual e adicioná-la à rede virtual criada.

Depois que a VM for criada, será possível usar o SSH no computador e instalar a ferramenta de linha de comando [psql](#).

Com o psql, conecte-se usando o comando abaixo. Substitua os valores por um nome do servidor e uma senha que sejam reais.

```
psql -h mydemoserver.postgres.database.azure.com -u mydemouser -p
```

NOTE

Se você receber um erro `The parameter PrivateDnsZoneArguments is required, and must be provided by customer`, isso significa que pode estar executando uma versão mais antiga da CLI do Azure. [Atualize a CLI do Azure](#) e repita a operação.

Limpar os recursos

Se não precisar desses recursos para outro início rápido/tutorial, você poderá excluí-los ao fazer o seguinte comando:

```
az group delete --name myresourcegroup
```

Caso queira apenas excluir o servidor recém-criado, será possível executar o comando

```
az postgres flexible-server delete .
```

```
az postgres flexible-server delete --resource-group myresourcegroup --name mydemoserver
```

Próximas etapas

[Implantar um aplicativo Django usando o Serviço de Aplicativo e o PostgreSQL](#)

Início Rápido: Usar um modelo ARM para criar um Banco de Dados do Azure para PostgreSQL – Servidor Flexível

01/07/2021 • 4 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O servidor flexível é um serviço gerenciado usado para executar, gerenciar e dimensionar bancos de dados do PostgreSQL altamente disponíveis na nuvem. Será possível usar um modelo do Azure Resource Manager (modelo do ARM) para provisionar um Servidor Flexível do PostgreSQL com o objetivo de implantar vários servidores ou diversos bancos de dados em um servidor.

Um [modelo ARM](#) é um arquivo JSON (JavaScript Object Notation) que define a infraestrutura e a configuração do projeto. O modelo usa a sintaxe declarativa. Na sintaxe declarativa, você descreve a implantação pretendida sem gravar a sequência de comandos de programação para criar a implantação.

O Azure Resource Manager é o serviço de implantação e gerenciamento do Azure. Ele fornece uma camada de gerenciamento que lhe permite criar, atualizar e excluir recursos em sua conta do Azure. Use recursos de gerenciamento, como controle de acesso, bloqueios e marcas, para proteger e organizar seus recursos após a implantação. Para saber mais sobre os modelos do Azure Resource Manager, confira [Visão geral de implantação de modelo](#).

Pré-requisitos

Uma conta do Azure com uma assinatura ativa. [Crie um gratuitamente.](#)

Examinar o modelo

Um Servidor do Banco de Dados do Azure para PostgreSQL é o recurso pai de um ou mais bancos de dados de uma região. Ele fornece o escopo das políticas de gerenciamento que se aplicam aos respectivos bancos de dados: logon, firewall, usuários, funções, configurações etc.

Crie um arquivo *postgres-flexible-server-template.json* e copie o script JSON a seguir nele.

```
{  
  "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json#",  
  "contentVersion": "1.0.0.0",  
  "parameters": {  
    "administratorLogin": {  
      "type": "String"  
    },  
    "administratorLoginPassword": {  
      "type": "SecureString"  
    },  
    "location": {  
      "type": "String"  
    },  
    "serverName": {  
      "type": "String"  
    }  
  },  
  "resources": [  
    {  
      "name": "flexibleServer",  
      "type": "Microsoft.DBforPostgreSQL/flexibleServers",  
      "location": "[parameters('location')]",  
      "tags": {},  
      "properties": {  
        "administratorLogin": "[parameters('administratorLogin')]",  
        "administratorLoginPassword": "[parameters('administratorLoginPassword')]",  
        "location": "[parameters('location')]",  
        "name": "[parameters('serverName')]"  
      }  
    }  
  ]  
}
```

```

    },
    "serverEdition": {
        "type": "String"
    },
    "storageSizeMB": {
        "type": "Int"
    },
    "haEnabled": {
        "type": "string"
    },
    "availabilityZone": {
        "type": "String"
    },
    "version": {
        "type": "String"
    },
    "tags": {
        "defaultValue": {},
        "type": "Object"
    },
    "firewallRules": {
        "defaultValue": {},
        "type": "Object"
    },
    "vnetData": {
        "defaultValue": {},
        "type": "Object"
    },
    "backupRetentionDays": {
        "type": "Int"
    }
},
"variables": {
    "api": "2020-02-14-privatepreview",
    "firewallRules": "[parameters('firewallRules').rules]",
    "publicNetworkAccess": "[if(empty(parameters('vnetData')), 'Enabled', 'Disabled')]",
    "vnetDataSet": "[if(empty(parameters('vnetData')), json('{ \"subnetArmResourceId\": \"\" }'), parameters('vnetData'))]",
    "finalVnetData": "[json(concat('{ \"subnetArmResourceId\": \"', variables('vnetDataSet').subnetArmResourceId, '\"}'))]"
},
"resources": [
{
    "type": "Microsoft.DBforPostgreSQL/flexibleServers",
    "apiVersion": "[variables('api')]",
    "name": "[parameters('serverName')]",
    "location": "[parameters('location')]",
    "sku": {
        "name": "Standard_D4ds_v4",
        "tier": "[parameters('serverEdition')]"
    },
    "tags": "[parameters('tags')]",
    "properties": {
        "version": "[parameters('version')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "[parameters('administratorLoginPassword')]",
        "publicNetworkAccess": "[variables('publicNetworkAccess')]",
        "DelegatedSubnetArguments": "[if(empty(parameters('vnetData')), json('null'), variables('finalVnetData'))]",
        "haEnabled": "[parameters('haEnabled')]",
        "storageProfile": {
            "storageMB": "[parameters('storageSizeMB')]",
            "backupRetentionDays": "[parameters('backupRetentionDays')]"
        },
        "availabilityZone": "[parameters('availabilityZone')]"
    }
},
{
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2019-08-01"
}
]

```

```

    "apiVersion": "2019-06-01",
    "name": "[concat('firewallRules-', copyIndex())]",
    "dependsOn": [
        "[concat('Microsoft.DBforPostgreSQL/flexibleServers/', parameters('serverName'))]"
    ],
    "properties": {
        "mode": "Incremental",
        "template": {
            "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json#",
            "contentVersion": "1.0.0.0",
            "resources": [
                {
                    "type": "Microsoft.DBforPostgreSQL/flexibleServers/firewallRules",
                    "name": "[concat(parameters('serverName'), '/', variables('firewallRules')[copyIndex()].name)]",
                    "apiVersion": "[variables('api')]",
                    "properties": {
                        "StartIpAddress": "[variables('firewallRules')[copyIndex()].startIPAddress]",
                        "EndIpAddress": "[variables('firewallRules')[copyIndex()].endIPAddress]"
                    }
                }
            ]
        },
        "copy": {
            "name": "firewallRulesIterator",
            "count": "[if(greater(length(variables('firewallRules')), 0), length(variables('firewallRules')), 1)]",
            "mode": "Serial"
        },
        "condition": "[greater(length(variables('firewallRules')), 0)]"
    }
}
]
}

```

Esses recursos do Azure estão definidos no modelo:

- Microsoft.DBforPostgreSQL/flexibleServers

Implantar o modelo

Selecione **Experimentar** no seguinte bloco de código do PowerShell para abrir o Azure Cloud Shell.

```

$serverName = Read-Host -Prompt "Enter a name for the new Azure Database for PostgreSQL server"
$resourceGroupName = Read-Host -Prompt "Enter a name for the new resource group where the server will exist"
.setLocation = Read-Host -Prompt "Enter an Azure region (for example, centralus) for the resource group"
$adminUser = Read-Host -Prompt "Enter the Azure Database for PostgreSQL server's administrator account name"
$adminPassword = Read-Host -Prompt "Enter the administrator password" -AsSecureString

New-AzResourceGroup -Name $resourceGroupName -Location $location # Use this command when you need to create
a new resource group for your deployment
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName `

    -TemplateFile "D:\Azure\Templates\EngineeringSite.json"
    -serverName $serverName `
    -administratorLogin $adminUser `
    -administratorLoginPassword $adminPassword

Read-Host -Prompt "Press [ENTER] to continue ..."

```

Examinar os recursos implantados

Siga estas etapas para verificar se o servidor foi criado no Azure.

- [Azure portal](#)
- [PowerShell](#)
- [CLI](#)

1. No [portal do Azure](#), pesquise por **Servidores Flexíveis do Banco de Dados do Azure para PostgreSQL (Versão Prévia)** e selecione-os.
2. Na lista de banco de dados, selecione o novo servidor para exibir a página **Visão Geral** com o objetivo de gerenciar o servidor.

Limpar os recursos

Mantenha esse grupo de recursos, o servidor e o banco de dados individual caso deseje ir para as [Próximas etapas](#). As próximas etapas mostram como se conectar e consultar seu banco de dados usando diferentes métodos.

Para excluir o grupo de recursos:

- [Portal](#)
- [PowerShell](#)
- [CLI do Azure](#)

No [portal](#), selecione o grupo de recursos que você deseja excluir.

1. Selecione **Excluir grupo de recursos**.
2. Para confirmar a exclusão, digite o nome do grupo de recursos

Próximas etapas

[Como migrar seu banco de dados usando o despejo e a restauração](#)

Guia de início rápido: como se conectar ao Banco de Dados do Azure para PostgreSQL – Servidor Flexível e executar consultas nele usando a CLI do Azure

08/07/2021 • 4 minutes to read

IMPORTANT

O Banco de Dados do Azure para PostgreSQL – Servidor Flexível atualmente está em versão prévia pública.

Este guia de início rápido demonstrará de que modo se conectar a um Servidor Flexível do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure com o `az postgres flexible-server connect` e executar a consulta única ou o arquivo SQL com o comando `az postgres flexible-server execute`. Esse comando permitirá testar a conectividade com o servidor de banco de dados e executar consultas. Também será possível executar várias consultas usando um modo interativo.

Pré-requisitos

- Uma conta do Azure. Se você não tiver uma, [obtenha uma avaliação gratuita](#).
- Instalar a versão mais recente da [CLI do Azure](#) (2.20.0 ou posterior)
- Fazer logon usando a CLI do Azure com o comando `az login`
- Ativar a persistência de parâmetro usando o comando `az config param-persist on`. A persistência de parâmetro ajudará você a usar o contexto local sem precisar repetir vários argumentos, como o grupo de recursos ou o local.

Criar um Servidor Flexível do PostgreSQL

A primeira coisa que criaremos é um servidor PostgreSQL gerenciado. No [Azure Cloud Shell](#), execute o script a seguir e anote o **nome do servidor**, o **nome de usuário** e a **senha** gerados desse comando.

```
az postgres flexible-server create --public-access <your-ip-address>
```

É possível fornecer argumentos adicionais para esse comando a fim de personalizá-lo. Confira todos os argumentos do comando [az postgres flexible-server create](#).

Exibir todos os argumentos

É possível exibir todos os argumentos desse comando usando o argumento `--help`.

```
az postgres flexible-server connect --help
```

Testar a conexão do servidor de banco de dados

É possível testar e validar a conexão com o banco de dados do ambiente de desenvolvimento usando o

comando.

```
az postgres flexible-server connect -n <servername> -u <username> -p "<password>" -d <dbname>
```

Exemplo:

```
az postgres flexible-server connect -n postgresdemoserver -u dbuser -p "dbpassword" -d postgres
```

Você verá a saída caso a conexão tenha sido estabelecida com êxito.

```
Command group 'postgres flexible-server' is in preview and under development. Reference and support levels:  
https://aka.ms/CLI\_refstatus  
Successfully connected to postgresdemoserver.  
Local context is turned on. Its information is saved in working directory C:\mydir. You can run `az local-context off` to turn it off.  
Your preference of  are now saved to local context. To learn more, type in `az local-context --help`
```

Caso haja falha na conexão, tente usar estas soluções:

- Verifique se a porta 5432 está aberta no computador cliente.
- Verifique se o nome de usuário e a senha do administrador do servidor estão corretos
- Verifique se a regra de firewall foi configurada para o computador cliente
- Caso tenha configurado o servidor com acesso privado na rede virtual, verifique se o computador cliente está na mesma rede virtual.

Executar várias consultas usando um modo interativo

É possível executar várias consultas usando um modo **interativo**. Execute o comando a seguir para habilitar o modo interativo

```
az postgres flexible-server connect -n <servername> -u <username> -p "<password>" -d <dbname>
```

Exemplo:

```
az postgres flexible-server connect -n postgresdemoserver -u dbuser -p "dbpassword" -d flexibleserverdb --interactive
```

Você verá a experiência do shell do **psql**, conforme mostrado abaixo:

```
Command group 'postgres flexible-server' is in preview and under development. Reference and support levels:  
https://aka.ms/CLI\_refstatus  
Password for earthyTurtle7:  
Server: PostgreSQL 12.5  
Version: 3.0.0  
Chat: https://gitter.im/dbcli/pgcli  
Home: http://pgcli.com  
postgres> create database pollsdb;  
CREATE DATABASE  
Time: 0.308s  
postgres> exit  
Goodbye!  
Local context is turned on. Its information is saved in working directory C:\sunita. You can run `az local-context off` to turn it off.  
Your preference of  are now saved to local context. To learn more, type in `az local-context --help`
```

Executar uma consulta única

É possível executar uma consulta única usando o comando com o argumento `--querytext`, `-q`.

```
az postgres flexible-server execute -n <server-name> -u <username> -p "<password>" -d <database-name> -q "<query-text>"
```

Exemplo:

```
az postgres flexible-server execute -n postgresdemoserver -u dbuser -p "dbpassword" -d flexibleserverdb -q "select * from table1;" --output table
```

Você verá uma saída semelhante ao mostrado abaixo:

```
Command group 'postgres flexible-server' is in preview and under development. Reference and support levels:  
https://aka.ms/CLI\_refstatus  
Successfully connected to postgresdemoserver.  
Ran Database Query: 'select * from table1;'  
Retrieving first 30 rows of query output, if applicable.  
Closed the connection postgresdemoserver.  
Local context is turned on. Its information is saved in working directory C:\mydir. You can run `az local-context off` to turn it off.  
Your preference of  are now saved to local context. To learn more, type in `az local-context --help`  
Txt      Val  
-----  
test    200  
test    200  
test    200  
test    200  
test    200  
test    200  
test    200
```

Executar arquivo SQL

É possível executar um arquivo SQL usando o comando com o argumento `--file-path`, `-f`.

```
az postgres flexible-server execute -n <server-name> -u <username> -p "<password>" -d <database-name> --file-path "<file-path>"
```

Exemplo:

```
az postgres flexible-server execute -n postgresdemoserver -u dbuser -p "dbpassword" -d flexibleserverdb -f "./test.sql"
```

Você verá uma saída semelhante ao mostrado abaixo:

```
Command group 'postgres flexible-server' is in preview and under development. Reference and support levels:  
https://aka.ms/CLI\_refstatus  
Running sql file '.\test.sql'...  
Successfully executed the file.  
Closed the connection to postgresdemoserver.
```

Próximas etapas

[Gerenciar o servidor](#)

Início Rápido: Usar o Python para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

21/05/2021 • 5 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Neste guia de início rápido, será possível se conectar a um Servidor Flexível do Banco de Dados do Azure para PostgreSQL usando o Python. Você usará instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados de plataformas Windows, Ubuntu Linux e Mac.

Este artigo pressupõe que você está familiarizado com o desenvolvimento usando Python, mas começou recentemente a trabalhar com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Um Banco de Dados do Azure para PostgreSQL – Servidor Flexível. Para criar um servidor flexível, confira [Criar um Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando o portal do Azure](#).
- [Python](#) 2.7 ou 3.6+.
- Instalador do pacote [pip](#) mais recente.

Como preparar sua estação de trabalho cliente

- Caso tenha criado um servidor flexível com *acesso privado* (*Integração VNet*), será necessário se conectar ao servidor de um recurso na mesma VNet do servidor. Crie uma máquina virtual e adicione-a à VNET criada com o servidor flexível. Confira [Criar e gerenciar a rede virtual do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a CLI do Azure](#).
- Caso tenha criado um servidor flexível com *acesso público* (*endereços IP permitidos*), será possível adicionar seu endereço IP local à lista de regras de firewall no servidor. Confira [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a CLI do Azure](#).

Instalar as bibliotecas do Python para PostgreSQL

O módulo [psycopg2](#) permite estabelecer conexão e consultar um banco de dados PostgreSQL e está disponível como um pacote [indicador](#) do Linux, macOS ou Windows. Instale a versão binária do módulo, incluindo todas as dependências. Para saber mais sobre instalação e requisitos de [psycopg2](#), confira [Instalação](#).

Para instalar [psycopg2](#), abra um terminal ou um prompt de comando e execute o comando
`pip install psycopg2`.

Obter informações da conexão de banco de dados

A conexão a Banco de Dados do Azure para PostgreSQL – Servidor Flexível requer o nome do servidor totalmente qualificado e as credenciais de logon. Você pode obter essas informações no portal do Azure.

1. No [portal do Azure](#), pesquise e selecione o nome do servidor flexível.
2. Na página Visão Geral do servidor, copie o **Nome do servidor** totalmente qualificado e o **Nome de usuário do administrador**. O **Nome do servidor** totalmente qualificado está sempre no formato `<my-server-name>.postgres.database.azure.com`.

Você também precisa da sua senha de administrador. Se você se esquecer dele, poderá redefini-lo na página de visão geral.

Como executar os exemplos de Python

Para cada exemplo de código neste artigo:

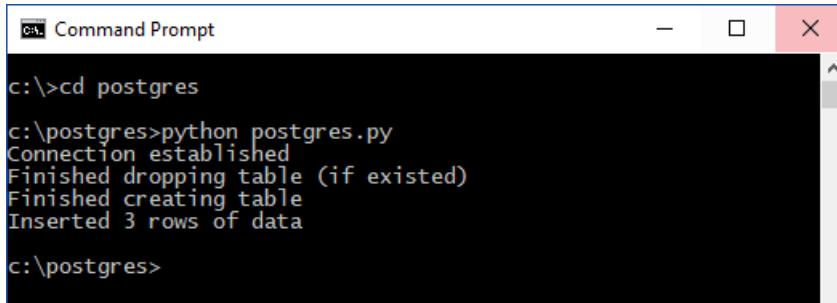
1. Crie um arquivo em um editor de texto.
2. Adicione o exemplo de código ao arquivo. No código, substitua:
 - `<server-name>` e `<admin-username>` pelos valores copiados do portal do Azure.
 - `<admin-password>` pela sua senha de servidor.
 - `<database-name>` pelo nome do banco de dados referente ao Banco de Dados do Azure para PostgreSQL – Servidor Flexível. Um banco de dados padrão chamado *postgres* foi criado automaticamente quando você criou seu servidor. É possível renomear esse banco de dados ou criar outro usando os comandos SQL.
3. Salve o arquivo na pasta do seu projeto com uma extensão `.py`, como `postgres-insert.py`. Para Windows, verifique se a codificação UTF-8 está selecionada quando você salvar o arquivo.
4. Para executar o arquivo, altere a pasta do projeto em um interface de linha de comando e digite `python` seguido pelo nome do arquivo, por exemplo `python postgres-insert.py`.

Criar uma tabela e inserir dados

O exemplo de código a seguir conecta-se ao seu banco de dados referente ao Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a função `psycopg2.connect` e carrega os dados com uma instrução `INSERT`. A função `cursor.execute` executa a consulta SQL no banco de dados.

```
import psycopg2
# Update connection string information
host = "<server-name>"
dbname = "<database-name>"
user = "<admin-username>"
password = "<admin-password>"
sslmode = "require"
# Construct connection string
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password,
sslmode)
conn = psycopg2.connect(conn_string)
print("Connection established")
cursor = conn.cursor()
# Drop previous table of same name if one exists
cursor.execute("DROP TABLE IF EXISTS inventory;")
print("Finished dropping table (if existed)")
# Create a table
cursor.execute("CREATE TABLE inventory (id serial PRIMARY KEY, name VARCHAR(50), quantity INTEGER);")
print("Finished creating table")
# Insert some data into the table
cursor.execute("INSERT INTO inventory (name, quantity) VALUES (%s, %s);", ("banana", 150))
cursor.execute("INSERT INTO inventory (name, quantity) VALUES (%s, %s);", ("orange", 154))
cursor.execute("INSERT INTO inventory (name, quantity) VALUES (%s, %s);", ("apple", 100))
print("Inserted 3 rows of data")
# Clean up
conn.commit()
cursor.close()
conn.close()
```

Quando o código é executado com êxito, ele produz a seguinte saída:



```
c:\ Command Prompt
c:\>cd postgres
c:\postgres>python postgres.py
Connection established
Finished dropping table (if existed)
Finished creating table
Inserted 3 rows of data
c:\postgres>
```

Ler dados

O exemplo de código a seguir conecta-se ao seu banco de dados referente ao Banco de Dados do Azure para PostgreSQL – Servidor Flexível e usa `cursor.execute` com a instrução `SELECT` do SQL para leitura de dados. Essa função aceita uma consulta e retorna um conjunto de resultados a ser iterado usando `cursor.fetchall()`.

```

import psycopg2
# Update connection string information
host = "<server-name>"
dbname = "<database-name>"
user = "<admin-username>"
password = "<admin-password>"
sslmode = "require"
# Construct connection string
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password,
sslmode)
conn = psycopg2.connect(conn_string)
print("Connection established")
cursor = conn.cursor()
# Fetch all rows from table
cursor.execute("SELECT * FROM inventory;")
rows = cursor.fetchall()
# Print all rows
for row in rows:
    print("Data row = (%s, %s, %s)" %(str(row[0]), str(row[1]), str(row[2])))
# Cleanup
conn.commit()
cursor.close()
conn.close()

```

Atualizar dados

O exemplo de código a seguir conecta-se ao seu banco de dados referente ao Banco de Dados do Azure para PostgreSQL – Servidor Flexível e usa `cursor.execute` com a declaração de **UPDATE** do SQL para atualização de dados.

```

import psycopg2
# Update connection string information
host = "<server-name>"
dbname = "<database-name>"
user = "<admin-username>"
password = "<admin-password>"
sslmode = "require"
# Construct connection string
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password,
sslmode)
conn = psycopg2.connect(conn_string)
print("Connection established")
cursor = conn.cursor()
# Update a data row in the table
cursor.execute("UPDATE inventory SET quantity = %s WHERE name = %s;", (200, "banana"))
print("Updated 1 row of data")
# Cleanup
conn.commit()
cursor.close()
conn.close()

```

Excluir dados

O exemplo de código a seguir conecta-se ao seu banco de dados referente ao Banco de Dados do Azure para PostgreSQL – Servidor Flexível e usa `cursor.execute` com a instrução **DELETE** do SQL para exclusão de um item de inventário que você inseriu anteriormente.

```
import psycopg2
# Update connection string information
host = "<server-name>"
dbname = "<database-name>"
user = "<admin-username>"
password = "<admin-password>"
sslmode = "require"
# Construct connection string
conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password,
sslmode)
conn = psycopg2.connect(conn_string)
print("Connection established")
cursor = conn.cursor()
# Delete data row from table
cursor.execute("DELETE FROM inventory WHERE name = %s;", ("orange",))
print("Deleted 1 row of data")
# Cleanup
conn.commit()
cursor.close()
conn.close()
```

Próximas etapas

[Como migrar seu banco de dados usando o despejo e a restauração](#)

Início rápido: Usar o Java e o JDBC com o Servidor Flexível do Banco de Dados do Azure para PostgreSQL

11/08/2021 • 10 minutes to read

Este tópico demonstra como criar um aplicativo de exemplo que usa o Java e o [JDBC](#) para armazenar e recuperar informações no [Servidor Flexível do Banco de Dados do Azure para PostgreSQL](#).

O JDBC é a API Java padrão para se conectar a bancos de dados relacionais tradicionais.

Pré-requisitos

- Uma conta do Azure. Se você não tiver uma, [obtenha uma avaliação gratuita](#).
- [Azure Cloud Shell](#) ou [CLI do Azure](#). É recomendável usar o Azure Cloud Shell para que o logon seja feito automaticamente e você tenha acesso a todas as ferramentas necessárias.
- Um [Java Development Kit](#) compatível, versão 8 (incluído no Azure Cloud Shell).
- A ferramenta de build [Apache Maven](#).

Preparar o ambiente de trabalho

Vamos usar as variáveis de ambiente para limitar erros de digitação e facilitar a personalização da configuração a seguir para as suas necessidades específicas.

Configure essas variáveis de ambiente usando os seguintes comandos:

```
AZ_RESOURCE_GROUP=database-workshop
AZ_DATABASE_NAME=<YOUR_DATABASE_NAME>
AZ_LOCATION=<YOUR_AZURE_REGION>
AZ_POSTGRESQL_USERNAME=demo
AZ_POSTGRESQL_PASSWORD=<YOUR_POSTGRESQL_PASSWORD>
AZ_LOCAL_IP_ADDRESS=<YOUR_LOCAL_IP_ADDRESS>
```

Substitua os espaços reservados pelos seguintes valores, que são usados em todo este artigo:

- <YOUR_DATABASE_NAME> : O nome do servidor PostgreSQL. O nome deve ser exclusivo em todo o Azure.
- <YOUR_AZURE_REGION> : A região do Azure que você usará. Você pode usar `eastus` por padrão, mas é recomendável configurar uma região mais próxima de onde você mora. Você pode ter a lista completa de regiões disponíveis ao digitar `az account list-locations`.
- <YOUR_POSTGRESQL_PASSWORD> : A senha do servidor de banco de dados PostgreSQL. Essa senha deveria ter um mínimo de oito caracteres. Os caracteres deveriam ser de três das seguintes categorias: Letras maiúsculas, letras minúsculas, números (0-9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
- <YOUR_LOCAL_IP_ADDRESS> : O endereço IP do computador local, no qual você executará o aplicativo Java. Uma forma conveniente de encontrá-lo é apontar o navegador para whatsmyip.akamai.com.

Em seguida, crie um grupo de recursos usando o seguinte comando:

```
az group create \
--name $AZ_RESOURCE_GROUP \
--location $AZ_LOCATION \
| jq
```

NOTE

Usamos o utilitário `jq` para exibir os dados JSON e torná-los mais legíveis. Esse utilitário é instalado por padrão no [Azure Cloud Shell](#). Se você não gostar desse utilitário, poderá removê-lo com segurança a parte `| jq` de todos os comandos que usaremos.

Criar uma instância do Banco de Dados do Azure para PostgreSQL

A primeira coisa que criaremos é um servidor PostgreSQL gerenciado.

NOTE

Você pode ler informações mais detalhadas sobre como criar servidores PostgreSQL em [Criar um servidor do Banco de Dados do Azure para PostgreSQL usando o portal do Azure](#).

No [Azure Cloud Shell](#), execute o seguinte comando:

```
az postgres flexible-server create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_NAME \
--location $AZ_LOCATION \
--sku-name Standard_B1s \
--storage-size 5120 \
--admin-user $AZ_POSTGRESQL_USERNAME \
--admin-password $AZ_POSTGRESQL_PASSWORD \
--public-access $AZ_LOCAL_IP_ADDRESS \
| jq
```

[Está tendo algum problema? Fale conosco.](#)

Configurar um banco de dados PostgreSQL

O servidor PostgreSQL que você criou anteriormente tem um banco de dados `postgres` vazio. Para este início rápido, vamos criar um banco de dados chamado `demo` seguindo estas etapas:

1. Instale a instância local do `psql` para se conectar a um servidor PostgreSQL do Azure.
2. Execute o comando `psql` a seguir para se conectar a um banco de dados `postgres` em seu servidor PostgreSQL

```
psql --host=<servername> --port=<port> --username=<user> --dbname=postgres
```

Exemplo:

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin --dbname=postgres
```

3. Crie um banco de dados chamado `demo` no prompt digitando o seguinte comando:

```
CREATE DATABASE demo;
```

Está tendo algum problema? Fale conosco.

Criar um projeto Java

Usando o seu IDE favorito, crie um projeto Java e adicione um arquivo `pom.xml` no diretório raiz:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>

  <properties>
    <java.version>1.8</java.version>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.2.12</version>
    </dependency>
  </dependencies>
</project>
```

Esse arquivo é um [Apache Maven](#) que configura o projeto a ser usado:

- Java 8
- Um driver PostgreSQL recente para Java

Preparar um arquivo de configuração para se conectar ao Banco de Dados do Azure para PostgreSQL

Crie o arquivo `src/main/resources/application.properties` e adicione:

```
url=jdbc:postgresql://$AZ_DATABASE_NAME.postgres.database.azure.com:5432/demo?ssl=true&sslmode=require
user=demo
password=$AZ_POSTGRESQL_PASSWORD
```

- Substitua as duas variáveis `$AZ_DATABASE_NAME` pelo valor que você configurou no início deste artigo.
- Substitua a variável `$AZ_POSTGRESQL_PASSWORD` pelo valor que você configurou no início deste artigo.

NOTE

Acrescentamos `?ssl=true&sslmode=require` à propriedade de configuração `url` para instruir o driver JDBC a usar o [protocolo TLS](#) ao se conectar ao banco de dados. O uso do TLS com o Banco de Dados do Azure para PostgreSQL é obrigatório e uma boa prática de segurança.

Criar um arquivo SQL para gerar o esquema de banco de dados

Usaremos um arquivo `src/main/resources/schema.sql` para criar um esquema de banco de dados. Crie esse arquivo com o seguinte conteúdo:

```
DROP TABLE IF EXISTS todo;
CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255), details VARCHAR(4096), done BOOLEAN);
```

Codificar o aplicativo

Conectarse ao banco de dados

Em seguida, adicione o código Java que usará o JDBC para armazenar e recuperar dados do servidor PostgreSQL.

Crie um arquivo `src/main/java/DemoApplication.java` que contém:

```
package com.example.demo;

import java.sql.*;
import java.util.*;
import java.util.logging.Logger;

public class DemoApplication {

    private static final Logger log;

    static {
        System.setProperty("java.util.logging.SimpleFormatter.format", "[%4$-7s] %5$s %n");
        log =Logger.getLogger(DemoApplication.class.getName());
    }

    public static void main(String[] args) throws Exception {
        log.info("Loading application properties");
        Properties properties = new Properties();

        properties.load(DemoApplication.class.getClassLoader().getResourceAsStream("application.properties"));

        log.info("Connecting to the database");
        Connection connection = DriverManager.getConnection(properties.getProperty("url"), properties);
        log.info("Database connection test: " + connection.getCatalog());

        log.info("Create database schema");
        Scanner scanner = new
Scanner(DemoApplication.class.getClassLoader().getResourceAsStream("schema.sql"));
        Statement statement = connection.createStatement();
        while (scanner.hasNextLine()) {
            statement.execute(scanner.nextLine());
        }

        /*
        Todo todo = new Todo(1L, "configuration", "congratulations, you have set up JDBC correctly!", true);
        insertData(todo, connection);
        todo = readData(connection);
        todo.setDetails("congratulations, you have updated data!");
        updateData(todo, connection);
        deleteData(todo, connection);
        */

        log.info("Closing database connection");
        connection.close();
    }
}
```

Está tendo algum problema? Fale conosco.

Esse código Java usará os arquivos `application.properties` e `schema.sql` que criamos anteriormente para se conectar ao servidor PostgreSQL e criar um esquema que armazenará nossos dados.

Nesse arquivo, você pode ver que comentamos os métodos para inserir, ler, atualizar e excluir dados: codificaremos esses métodos no restante deste artigo e você poderá remover as marcas de comentários uma após a outra.

NOTE

As credenciais de banco de dados são armazenadas nas propriedades `user` e `password` no arquivo `application.properties`. Essas credenciais são usadas durante a execução de

```
DriverManager.getConnection(properties.getProperty("url"), properties);
```

, pois o arquivo de propriedades é passado como um argumento.

Agora, você pode executar esta classe principal com a sua ferramenta favorita:

- Usando o IDE, você deve clicar com o botão direito do mouse na classe `DemoApplication` e executá-la.
- Usando o Maven, você pode executar o aplicativo por meio do:

```
mvn exec:java -Dexec.mainClass="com.example.demo.DemoApplication".
```

O aplicativo deve se conectar ao Banco de Dados do Azure para PostgreSQL, criar um esquema de banco de dados e fechar a conexão, como você vê nos logs do console:

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Closing database connection
```

Criar uma classe de domínio

Crie uma classe Java `Todo`, ao lado da classe `DemoApplication` e adicione o seguinte código:

```

package com.example.demo;

public class Todo {

    private Long id;
    private String description;
    private String details;
    private boolean done;

    public Todo() {
    }

    public Todo(Long id, String description, String details, boolean done) {
        this.id = id;
        this.description = description;
        this.details = details;
        this.done = done;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getDetails() {
        return details;
    }

    public void setDetails(String details) {
        this.details = details;
    }

    public boolean isDone() {
        return done;
    }

    public void setDone(boolean done) {
        this.done = done;
    }

    @Override
    public String toString() {
        return "Todo{" +
            "id=" + id +
            ", description='" + description + '\'' +
            ", details='" + details + '\'' +
            ", done=" + done +
            '}';
    }
}

```

Essa classe é um modelo de domínio mapeado na tabela `todo` que você criou ao executar o script `schema.sql`.

Inserir dados no Banco de Dados do Azure para PostgreSQL

No arquivo `src/main/java/DemoApplication.java`, após o método principal, adicione o seguinte método para

inserir dados no banco de dados:

```
private static void insertData(Todo todo, Connection connection) throws SQLException {
    log.info("Insert data");
    PreparedStatement insertStatement = connection
        .prepareStatement("INSERT INTO todo (id, description, details, done) VALUES (?, ?, ?, ?);");

    insertStatement.setLong(1, todo.getId());
    insertStatement.setString(2, todo.getDescription());
    insertStatement.setString(3, todo.getDetails());
    insertStatement.setBoolean(4, todo.isDone());
    insertStatement.executeUpdate();
}
```

Agora você pode remover a marca de comentário das seguintes duas linhas no método `main`:

```
Todo todo = new Todo(1L, "configuration", "congratulations, you have set up JDBC correctly!", true);
insertData(todo, connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO] Loading application properties
[INFO] Connecting to the database
[INFO] Database connection test: demo
[INFO] Create database schema
[INFO] Insert data
[INFO] Closing database connection
```

Como ler dados do Banco de Dados do Azure para PostgreSQL

Vamos ler os dados inseridos anteriormente para validar se o nosso código funciona corretamente.

No arquivo `src/main/java/DemoApplication.java`, após o método `insertData`, adicione o seguinte método para ler dados do banco de dados:

```
private static Todo readData(Connection connection) throws SQLException {
    log.info("Read data");
    PreparedStatement readStatement = connection.prepareStatement("SELECT * FROM todo;");
    ResultSet resultSet = readStatement.executeQuery();
    if (!resultSet.next()) {
        log.info("There is no data in the database!");
        return null;
    }
    Todo todo = new Todo();
    todo.setId(resultSet.getLong("id"));
    todo.setDescription(resultSet.getString("description"));
    todo.setDetails(resultSet.getString("details"));
    todo.setDone(resultSet.getBoolean("done"));
    log.info("Data read from the database: " + todo.toString());
    return todo;
}
```

Agora você pode remover a marca de comentário da seguinte linha no método `main`:

```
todo = readData(connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have set up JDBC correctly!', done=true}
[INFO    ] Closing database connection
```

Como atualizar dados no Banco de Dados do Azure para PostgreSQL

Vamos atualizar os dados que inserimos anteriormente.

Ainda no arquivo `src/main/java/DemoApplication.java`, após o método `readData`, adicione o seguinte método para atualizar os dados no banco de dados:

```
private static void updateData(Todo todo, Connection connection) throws SQLException {
    log.info("Update data");
    PreparedStatement updateStatement = connection
        .prepareStatement("UPDATE todo SET description = ?, details = ?, done = ? WHERE id = ?");

    updateStatement.setString(1, todo.getDescription());
    updateStatement.setString(2, todo.getDetails());
    updateStatement.setBoolean(3, todo.isDone());
    updateStatement.setLong(4, todo.getId());
    updateStatement.executeUpdate();
    readData(connection);
}
```

Agora você pode remover a marca de comentário das seguintes duas linhas no método `main`:

```
todo.setDetails("congratulations, you have updated data!");
updateData(todo, connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have set up JDBC correctly!', done=true}
[INFO    ] Update data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have updated data!', done=true}
[INFO    ] Closing database connection
```

Como excluir dados no Banco de Dados do Azure para PostgreSQL

Por fim, vamos excluir os dados que inserimos anteriormente.

Ainda no arquivo `src/main/java/DemoApplication.java`, após o método `updateData`, adicione o seguinte método para excluir os dados no banco de dados:

```
private static void deleteData(Todo todo, Connection connection) throws SQLException {
    log.info("Delete data");
    PreparedStatement deleteStatement = connection.prepareStatement("DELETE FROM todo WHERE id = ?;");
    deleteStatement.setLong(1, todo.getId());
    deleteStatement.executeUpdate();
    readData(connection);
}
```

Agora você pode remover a marca de comentário da seguinte linha no método `main`:

```
deleteData(todo, connection);
```

A execução da classe principal deve produzir a seguinte saída:

```
[INFO ] Loading application properties
[INFO ] Connecting to the database
[INFO ] Database connection test: demo
[INFO ] Create database schema
[INFO ] Insert data
[INFO ] Read data
[INFO ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have set up JDBC correctly!', done=true}
[INFO ] Update data
[INFO ] Read data
[INFO ] Data read from the database: Todo{id=1, description='configuration', details='congratulations, you have updated data!', done=true}
[INFO ] Delete data
[INFO ] Read data
[INFO ] There is no data in the database!
[INFO ] Closing database connection
```

Limpar os recursos

Parabéns! Você criou um aplicativo Java que usa o JDBC para armazenar e recuperar dados do Banco de Dados do Azure para PostgreSQL.

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Próximas etapas

[Migre seu banco de dados usando Exportar e Importar](#)

Início Rápido: Usar o .NET (C#) para se conectar e consultar dados no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

21/05/2021 • 6 minutes to read

Este guia de início rápido demonstra como se conectar a um banco de dados do Azure para PostgreSQL usando aplicativo C#. Ele mostra como usar instruções SQL para consultar, inserir, atualizar e excluir dados no banco de dados. As etapas neste artigo pressupõem que você está familiarizado com o desenvolvimento usando C# e que começou recentemente a trabalhar com o Banco de Dados do Azure para PostgreSQL.

Pré-requisitos

Para este início rápido você precisa:

- Uma conta do Azure com uma assinatura ativa. [Crie uma conta gratuitamente](#).
- Criar um servidor flexível do Banco de Dados do Azure para PostgreSQL usando o [portal do Azure](#) ou a [CLI do Azure](#) se ainda não tiver um.
- Use o banco de dados *postgres* vazio disponível no servidor ou crie um [banco de dados](#).
- Instale o [.NET Framework](#) para sua plataforma (Windows, Ubuntu Linux ou macOS).
- Instale o [Visual Studio](#) para criar seu projeto.
- Instale o pacote NuGet [Npgsql](#) no Visual Studio.

Obter informações de conexão

Obtenha as informações de conexão necessárias para se conectar ao Banco de Dados do Azure para PostgreSQL. Você precisa das credenciais de logon e do nome do servidor totalmente qualificado.

1. Faça logon no [Portal do Azure](#).
2. No menu à esquerda no portal do Azure, clique em **Todos os recursos** e pesquise o servidor que você criou (como **mydemoserver**).
3. Clique no nome do servidor.
4. No painel **Visão Geral** do servidor, anote o **Nome do servidor** e **Nome de logon do administrador do servidor**. Se você esquecer sua senha, também poderá redefini-la nesse painel.

The screenshot shows the Azure portal interface for managing a PostgreSQL flexible server. The top navigation bar includes 'Search (Ctrl+)', 'Delete', 'Reset password', 'Restore', 'Restart', 'Stop', and 'Feedback' buttons. The main area has a left sidebar with 'Overview' (selected), 'Activity log', 'Access control (IAM)', 'Tags', and 'Settings' tabs. The 'Overview' tab displays details like 'Resource group: myproject', 'Status: Available', 'Location: East US', and 'Connectivity method: Public access (allowed IP addresses)'. On the right, there's a summary section with fields for 'Server name' (set to 'server174634945.postgres.database.azure.com') and 'Server admin login name' (set to 'fluidCod6'). Both of these fields are highlighted with red boxes.

Etapa 1: conectar e inserir dados

Use o código a seguir para se conectar e carregar os dados usando instruções SQL **CREATE TABLE** e **INSERT INTO**. O código usa a classe `NpgsqlCommand` com o método:

- `Open()` para estabelecer uma conexão com o banco de dados PostgreSQL.

- [CreateCommand\(\)](#) para definir a propriedade CommandText.
- [ExecuteNonQuery\(\)](#) para executar os comandos de banco de dados.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```
using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresCreate
    {
        // Obtain connection string information from the portal
        //

        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin";
        private static string DBname = "postgres";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //

            string connString =
                String.Format(
                    "Server={0};Username={1};Database={2};Port={3};Password={4};SSLMODE=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))

            {
                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("DROP TABLE IF EXISTS inventory", conn))
                {
                    command.ExecuteNonQuery();
                    Console.Out.WriteLine("Finished dropping table (if existed)");

                }

                using (var command = new NpgsqlCommand("CREATE TABLE inventory(id serial PRIMARY KEY, name
VARCHAR(50), quantity INTEGER)", conn))
                {
                    command.ExecuteNonQuery();
                    Console.Out.WriteLine("Finished creating table");
                }

                using (var command = new NpgsqlCommand("INSERT INTO inventory (name, quantity) VALUES (@n1,
@q1), (@n2, @q2), (@n3, @q3)", conn))
                {
                    command.Parameters.AddWithValue("n1", "banana");
                    command.Parameters.AddWithValue("q1", 150);
                    command.Parameters.AddWithValue("n2", "orange");
                    command.Parameters.AddWithValue("q2", 154);
                    command.Parameters.AddWithValue("n3", "apple");
                }
            }
        }
    }
}
```

```
        command.Parameters.AddWithValue("q3", 100);

        int nRows = command.ExecuteNonQuery();
        Console.Out.WriteLine(String.Format("Number of rows inserted={0}", nRows));
    }
}

Console.WriteLine("Press RETURN to exit");
Console.ReadLine();
}
}
```

Está tendo algum problema? Fale conosco.

Etapa 2: Ler dados

Use o código a seguir para conectar-se e ler os dados usando uma instrução SQL SELECT. O código usa a classe NpgsqlCommand com o método:

- [Open\(\)](#) para estabelecer uma conexão com o PostgreSQL.
- [CreateCommand\(\)](#) e [ExecuteReader\(\)](#) para executar os comandos de banco de dados.
- [Read\(\)](#) a fim de avançar para os registros nos resultados.
- [GetInt32\(\)](#) e [GetString\(\)](#) para analisar os valores do registro.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```

using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresRead
    {
        // Obtain connection string information from the portal
        //
        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin";
        private static string DBname = "postgres";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //
            string connString =
                String.Format(
                    "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSlMode=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))
            {

                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("SELECT * FROM inventory", conn))
                {

                    var reader = command.ExecuteReader();
                    while (reader.Read())
                    {
                        Console.WriteLine(
                            string.Format(
                                "Reading from table={({0}, {1}, {2})}",
                                reader.GetInt32(0).ToString(),
                                reader.GetString(1),
                                reader.GetInt32(2).ToString()
                            )
                        );
                    }
                    reader.Close();
                }
            }

            Console.WriteLine("Press RETURN to exit");
            Console.ReadLine();
        }
    }
}

```

Está tendo algum problema? Fale conosco.

Etapa 3: Atualizar dados

Use o código a seguir para conectar-se e atualizar os dados usando uma instrução SQL **UPDATE**. O código usa a

classe NpgsqlCommand com o método:

- [Open\(\)](#) para estabelecer uma conexão com o PostgreSQL.
- [CreateCommand\(\)](#) para definir a propriedade CommandText.
- [ExecuteNonQuery\(\)](#) para executar os comandos de banco de dados.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```
using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresUpdate
    {
        // Obtain connection string information from the portal
        //

        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin";
        private static string DBname = "postgres";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //

            string connString =
                String.Format(
                    "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSLMODE=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))
            {

                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("UPDATE inventory SET quantity = @q WHERE name = @n",
                conn))
                {
                    command.Parameters.AddWithValue("n", "banana");
                    command.Parameters.AddWithValue("q", 200);
                    int nRows = command.ExecuteNonQuery();
                    Console.Out.WriteLine(String.Format("Number of rows updated={0}", nRows));
                }
            }

            Console.WriteLine("Press RETURN to exit");
            Console.ReadLine();
        }
    }
}
```

Está tendo algum problema? Fale conosco.

Etapa 4: Excluir dados

Use o código a seguir para conectar-se e excluir os dados usando uma instrução SQL **DELETE**.

O código usa a classe `NpgsqlCommand` com o método `Open()` para estabelecer uma conexão com o banco de dados PostgreSQL. Em seguida, o código usa o método `CreateCommand()`, define a propriedade `CommandText` e chama o método `ExecuteNonQuery ()` para executar os comandos do banco de dados.

IMPORTANT

Substitua os parâmetros Host, DBName, User e Password pelos valores que você especificou quando criou o servidor e o banco de dados.

```

using System;
using Npgsql;

namespace Driver
{
    public class AzurePostgresDelete
    {
        // Obtain connection string information from the portal
        //

        private static string Host = "mydemoserver.postgres.database.azure.com";
        private static string User = "mylogin@mydemoserver";
        private static string DBname = "postgres";
        private static string Password = "<server_admin_password>";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from portal
            //
            string connString =
                String.Format(
                    "Server={0}; User Id={1}; Database={2}; Port={3}; Password={4};SSlMode=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password);

            using (var conn = new NpgsqlConnection(connString))
            {
                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new NpgsqlCommand("DELETE FROM inventory WHERE name = @n", conn))
                {
                    command.Parameters.AddWithValue("n", "orange");

                    int nRows = command.ExecuteNonQuery();
                    Console.Out.WriteLine(String.Format("Number of rows deleted={0}", nRows));
                }
            }

            Console.WriteLine("Press RETURN to exit");
            Console.ReadLine();
        }
    }
}

```

Limpar os recursos

Para limpar todos os recursos usados durante este guia de início rápido, exclua o grupo de recursos usando o seguinte comando:

```

az group delete \
--name $AZ_RESOURCE_GROUP \
--yes

```

Próximas etapas

[Gerenciar o servidor de Banco de Dados do Azure para MySQL usando o portal](#)

Tutorial: Implantar o aplicativo Django no AKS com o Banco de Dados do Azure para PostgreSQL – Servidor Flexível

21/05/2021 • 10 minutes to read

Neste guia de início rápido, você implanta um aplicativo Django no cluster AKS (Serviço de Kubernetes do Azure) com o Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão prévia) usando a CLI do Azure.

O [AKS](#) é um serviço de Kubernetes gerenciado que permite a implantação e o gerenciamento de clusters rapidamente. O [Banco de Dados do Azure para PostgreSQL – Servidor Flexível \(versão prévia\)](#) é um serviço de banco de dados totalmente gerenciado, projetado para proporcionar um controle mais granular e flexibilidade nas funções de gerenciamento de banco de dados e definições de configuração.

NOTE

- O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está atualmente na versão prévia pública
- Este guia de início rápido pressupõe uma compreensão básica dos conceitos do Kubernetes, do Django e do PostgreSQL.

Pré-requisitos

Se você não tiver uma [assinatura do Azure](#), crie uma [conta gratuita](#) antes de começar.

- Use o [Azure Cloud Shell](#) usando o ambiente bash.



- Se preferir, [instale](#) a CLI do Azure para executar comandos de referência da CLI.
 - Se estiver usando uma instalação local, entre com a CLI do Azure usando o comando [az login](#). Para concluir o processo de autenticação, siga as etapas exibidas no terminal. Confira [Entrar com a CLI do Azure](#) para obter outras opções de entrada.
 - Quando solicitado, instale as extensões da CLI do Azure no primeiro uso. Para obter mais informações sobre extensões, confira [Usar extensões com a CLI do Azure](#).
 - Execute [az version](#) para localizar a versão e as bibliotecas dependentes que estão instaladas. Para fazer a atualização para a versão mais recente, execute [az upgrade](#). Este artigo exige a versão mais recente da CLI do Azure. Se você está usando o Azure Cloud Shell, a versão mais recente já está instalada.

NOTE

Se estiver executando comandos neste início rápido localmente (em vez de no Azure Cloud Shell), execute-os como administrador.

Criar um grupo de recursos

Um grupo de recursos do Azure é um grupo lógico no qual os recursos do Azure são implantados e gerenciados. Vamos criar um grupo de recursos, *django-project* usando o comando [az group create][az-group-

`create]` na localização *eastus*.

```
az group create --name django-project --location eastus
```

NOTE

A localização do grupo de recursos é onde os metadados do grupo de recursos são armazenados. Também é onde seus recursos são executados no Azure, caso você não especifique outra região durante a criação de recursos.

A seguinte saída de exemplo mostra o grupo de recursos criado com êxito:

```
{
  "id": "/subscriptions/<guid>/resourceGroups/django-project",
  "location": "eastus",
  "managedBy": null,

  "name": "django-project",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

Criar cluster AKS

Use o comando `az aks create` para criar um cluster do AKS. O exemplo a seguir cria um cluster chamado *myAKSCluster* com um nó. Isso levará vários minutos.

```
az aks create --resource-group django-project --name djangoappcluster --node-count 1 --generate-ssh-keys
```

Após alguns minutos, o comando será concluído e retornará informações no formato JSON sobre o cluster.

NOTE

Durante a criação de um cluster do AKS, um segundo grupo de recursos é criado automaticamente para armazenar os recursos do AKS. Confira [Por que são criados dois grupos de recursos com o AKS?](#)

Conectar-se ao cluster

Para gerenciar um cluster do Kubernetes, use `kubectl`, o cliente de linha de comando do Kubernetes. Se você usar o Azure Cloud Shell, o `kubectl` já estará instalado. Para instalar o `kubectl` localmente, use o comando `az aks install-cli`:

```
az aks install-cli
```

Para configurar o `kubectl` para se conectar ao cluster do Kubernetes, use o comando `az aks get-credentials`. Este comando baixa as credenciais e configura a CLI do Kubernetes para usá-las.

```
az aks get-credentials --resource-group django-project --name djangoappcluster
```

NOTE

O comando acima usa a localização padrão para o [arquivo de configuração do Kubernetes](#), que é `~/.kube/config`. Especifique outra localização para o arquivo de configuração do Kubernetes usando `--file`.

Para verificar a conexão com o cluster, use o comando `kubectl get` para retornar uma lista dos nós de cluster.

```
kubectl get nodes
```

A saída de exemplo a seguir mostra o único nó criado nas etapas anteriores. Verifique se o status do nó é *Pronto*:

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-31718369-0	Ready	agent	6m44s	v1.12.8

Criar um Banco de Dados do Azure para PostgreSQL – Servidor Flexível

Crie um servidor flexível com o comando `az postgres flexible-server create`. O seguinte comando criará um servidor usando valores e padrões de serviço do contexto local da CLI do Azure:

```
az postgres flexible-server create --public-access <YOUR-IP-ADDRESS>
```

O servidor criado terá os atributos abaixo:

- Um banco de dados vazio, `postgres`, é criado quando o servidor é provisionado pela primeira vez. Neste início rápido, usaremos este banco de dados.
- Nome do servidor gerado automaticamente, nome de usuário do administrador, senha de administrador, nome do grupo de recursos (caso ainda não tenha sido especificado no contexto local). Além disso, o servidor e o grupo de recursos estarão na mesma localização
- Padrões de serviço para as configurações de servidor restantes: nível de computação (Uso Geral), tamanho da computação/SKU (Standard_D2s_v3 que usa 2vCores), período de retenção de backup (7 dias) e versão do PostgreSQL (12)
- O uso do argumento de acesso público permite que você crie um servidor com acesso público protegido por regras de firewall. Fornecendo seu endereço IP para adicionar a regra de firewall para permitir o acesso de seu computador cliente.
- Como o comando está usando o contexto local, ele criará o servidor no grupo de recursos `django-project` e na região `eastus`.

Criar sua imagem do Docker do Django

Crie um [aplicativo Django](#) ou use seu projeto Django existente. Verifique se o código está nesta estrutura de pastas.

```
└── my-djangoapp
    ├── views.py
    ├── models.py
    ├── forms.py
    └── templates
        ...
    └── static
        ...
    ...
└── my-django-project
    ├── settings.py
    ├── urls.py
    └── wsgi.py
    ...
    └── Dockerfile
    └── requirements.txt
    └── manage.py
```

Atualize `ALLOWED_HOSTS` no `settings.py` para verificar se o aplicativo Django usa o IP externo que é atribuído ao aplicativo Kubernetes.

```
ALLOWED_HOSTS = ['*']
```

Atualize a seção `DATABASES={ }` no arquivo `settings.py`. O snippet de código abaixo está lendo o host de banco de dados, o nome de usuário e a senha do arquivo de manifesto do Kubernetes.

```
DATABASES={
    'default':{
        'ENGINE':'django.db.backends.postgresql_psycopg2',
        'NAME':os.getenv('DATABASE_NAME'),
        'USER':os.getenv('DATABASE_USER'),
        'PASSWORD':os.getenv('DATABASE_PASSWORD'),
        'HOST':os.getenv('DATABASE_HOST'),
        'PORT':'5432',
        'OPTIONS': {'sslmode': 'require'}
    }
}
```

Gerar um arquivo requirements.txt

Crie um arquivo `requirements.txt` para listar as dependências para o aplicativo Django. Veja um arquivo `requirements.txt` de exemplo. Você pode usar `pip freeze > requirements.txt` para gerar um arquivo `requirements.txt` para seu aplicativo existente.

```
Django==2.2.17
postgres==3.0.0
psycopg2-binary==2.8.6
psycopg2-pool==1.1
pytz==2020.4
```

Criar um Dockerfile

Crie um arquivo chamado `Dockerfile` e copie o snippet de código abaixo. Esse Dockerfile na configuração do Python 3.8 e da instalação de todos os requisitos listados no arquivo `requirements.txt`.

```
# Use the official Python image from the Docker Hub
FROM python:3.8.2

# Make a new directory to put our code in.
RUN mkdir /code

# Change the working directory.
WORKDIR /code

# Copy to code folder
COPY . /code/

# Install the requirements.
RUN pip install -r requirements.txt

# Run the application:
CMD python manage.py runserver 0.0.0.0:8000
```

Crie sua imagem

Verifique se você está no diretório `my-django-app` em um terminal usando o comando `cd`. Execute o seguinte comando para criar sua imagem de painel de anúncios:

```
docker build --tag myblog:latest .
```

Implante sua imagem no [Docker Hub](#) ou no [Registro de Contêiner do Azure](#).

IMPORTANT

Se você estiver usando o ACR (Registro de Contêiner do Azure), execute o comando `az aks update` para anexar a conta do ACR ao cluster do AKS.

```
az aks update -n myAKSCluster -g django-project --attach-acr <your-acr-name>
```

Criar o arquivo de manifesto do Kubernetes

Um arquivo de manifesto do Kubernetes define um estado desejado para o cluster, como as imagens de contêiner a serem executadas. Vamos criar um arquivo de manifesto chamado `djangoapp.yaml` e copiar na definição YAML a seguir.

IMPORTANT

- Substitua `[DOCKER-HUB-USER/ACR ACCOUNT]/[YOUR-IMAGE-NAME]:[TAG]` pelo nome e pela marca da imagem do Docker do Django real, por exemplo, `docker-hub-user/myblog:latest`.
- Atualize a seção `env` abaixo com seu `SERVERNAME`, `YOUR-DATABASE-USERNAME` e `YOUR-DATABASE-PASSWORD` do seu servidor flexível postgres.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: django-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: django-app
  template:
    metadata:
      labels:
        app: django-app
    spec:
      containers:
        - name: django-app
          image: [DOCKER-HUB-USER-OR-ACR-ACCOUNT]/[YOUR-IMAGE-NAME]:[TAG]
          ports:
            - containerPort: 80
          env:
            - name: DATABASE_HOST
              value: "SERVERNAME.postgres.database.azure.com"
            - name: DATABASE_USERNAME
              value: "YOUR-DATABASE-USERNAME"
            - name: DATABASE_PASSWORD
              value: "YOUR-DATABASE-PASSWORD"
            - name: DATABASE_NAME
              value: "postgres"
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: "app"
                    operator: In
                    values:
                      - django-app
          topologyKey: "kubernetes.io/hostname"
---
apiVersion: v1
kind: Service
metadata:
  name: python-svc
spec:
  type: LoadBalancer
  ports:
    - port: 8000
  selector:
    app: django-app

```

Implantar Django no cluster AKS

Implante o aplicativo usando o comando `kubectl apply` e especifique o nome do manifesto YAML:

```
kubectl apply -f djangoapp.yaml
```

A seguinte saída de exemplo mostra as Implantações e os Serviços criados com êxito:

```
deployment "django-app" created
service "python-svc" created
```

Uma implantação `django-app` permite que você descreva detalhes de sua implantação, como quais imagens

usam para o aplicativo, o número de pods e a configuração de pod. Um serviço `python-svc` é criado para expor o aplicativo por meio de um IP externo.

Testar o aplicativo

Quando o aplicativo é executado, um serviço de Kubernetes expõe o front-end do aplicativo à Internet. A conclusão desse processo pode levar alguns minutos.

Para monitorar o andamento, use o comando `kubectl get service` com o argumento `--watch`.

```
kubectl get service django-app --watch
```

Inicialmente, o *EXTERNAL-IP* para o serviço *djangapp* de exemplo é mostrado como *pendente*.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
django-app	LoadBalancer	10.0.37.27	<pending>	80:30572/TCP	6s

Quando o endereço *EXTERNAL-IP* for alterado de *pendente* para um endereço IP público real, use `CTRL-C` para interromper o processo de inspeção do `kubectl`. A seguinte saída de exemplo mostra um endereço IP público válido atribuído ao serviço:

```
djangapp LoadBalancer 10.0.37.27 52.179.23.131 80:30572/TCP 2m
```

Agora abra um navegador da Web para o endereço IP externo de sua exibição de serviço do aplicativo Django.

NOTE

- No momento, o site do Django não está usando HTTPS. É recomendável [HABILITAR o TLS com os próprios certificados](#).
- Você pode habilitar o [roteamento HTTP](#) para o cluster. Quando o roteamento http está habilitado, ele configura um controlador de entrada em seu cluster AKS. À medida que os aplicativos são implantados, a solução também cria nomes DNS publicamente acessíveis para os terminais de aplicativos.

Executar migrações de banco de dados

Para qualquer aplicativo Django, você precisaria executar uma migração de banco de dados ou coletar arquivos estáticos. Você pode executar esses comandos do shell Django usando `$ kubectl exec <pod-name> -- [COMMAND]`. Antes de executar o comando, você precisa encontrar o nome do pod usando `kubectl get pods`.

```
$ kubectl get pods
```

Você verá uma saída como esta

NAME	READY	STATUS	RESTARTS	AGE
djangapp-5d9cd6cd8-16x4b	1/1	Running	0	2m

Depois que o nome do pod for encontrado, você poderá executar migrações de banco de dados Django com o comando `$ kubectl exec <pod-name> -- [COMMAND]`. Observe que `/code/` é o diretório de trabalho para o projeto definir no `Dockerfile` acima.

```
$ kubectl exec django-app-5d9cd6cd8-16x4b -- python /code/manage.py migrate
```

A saída teria uma aparência como

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  . . . . .
```

Se você encontrar problemas, execute `kubectl logs <pod-name>` para ver qual exceção é gerada pelo seu aplicativo. Se o aplicativo estiver funcionando com êxito, você verá uma saída como esta ao executar `kubectl logs`.

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply the migrations for
app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 08, 2020 - 23:24:14
Django version 2.2.17, using settings 'django_postgres_app.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Limpar os recursos

Para evitar cobranças do Azure, limpe recursos desnecessários. Quando o cluster não for mais necessário, use o comando [az group delete](#) para remover o grupo de recursos, o serviço de contêiner e todos os recursos relacionados.

```
az group delete --name django-project --yes --no-wait
```

NOTE

Quando você excluir o cluster, a entidade de serviço do Azure Active Directory usada pelo cluster do AKS não será removida. Para obter as etapas para remover a entidade de serviço, confira [Considerações sobre a entidade de serviço do AKS e sua exclusão](#). Se você tiver usado uma identidade gerenciada, ela será gerenciada pela plataforma e não exigirá remoção.

Próximas etapas

- Saiba como [acessar o painel da Web do Kubernetes](#) para seu cluster do AKS
- Saiba como [habilitar a implantação contínua](#)
- Saiba como [escalar seu cluster](#)
- Saiba como gerenciar seu [servidor flexível postgres](#)
- Saiba como [configurar parâmetros de servidor](#) para seu servidor de banco de dados.

Tutorial: Criar um Banco de Dados do Azure para PostgreSQL – Servidor Flexível com aplicativo Web dos Serviços de Aplicativos na rede virtual

21/05/2021 • 4 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este tutorial mostra como criar um aplicativo Web do Serviço de Aplicativo do Azure com o Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão prévia) dentro de uma [Rede virtual](#).

Neste tutorial, você aprenderá a:

- Criar um servidor flexível PostgreSQL em uma rede virtual
- Criar uma sub-rede a ser delegada ao Serviço de Aplicativo
- Criar um aplicativo Web
- Adicionar o aplicativo Web à rede virtual
- Conectar-se ao Postgres do aplicativo Web

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Este artigo exige que você esteja executando a CLI do Azure versão 2.0 ou posterior localmente. Para ver a versão instalada, execute o comando `az --version`. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Você precisará fazer logon em sua conta usando o comando [login az](#). Observe a propriedade **id** da saída do comando para o nome da assinatura correspondente.

```
az login
```

Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Selecione a ID da assinatura específica em sua conta usando o comando [az account set](#). Substitua a propriedade **ID da assinatura** da saída [az logon](#) por sua assinatura no espaço reservado da ID de assinatura.

```
az account set --subscription <subscription ID>
```

Criar um Servidor Flexível PostgreSQL em uma nova rede virtual

Crie um servidor flexível privado dentro de uma VNET (rede virtual) usando o seguinte comando:

```
az postgres flexible-server create --resource-group myresourcegroup --location westus2
```

Esse comando executa as seguintes ações, que podem levar alguns minutos:

- Crie o grupo de recursos se ele ainda não existir.
- Gera um nome do servidor, caso não tenha sido fornecido.
- Crie uma rede virtual para seu novo servidor PostgreSQL. Anote o nome da rede virtual e o nome da sub-rede criados para o servidor, pois você precisa adicionar o aplicativo Web à mesma rede virtual.
- Cria o nome de usuário e senha do administrador para o servidor, caso não fornecidos.
- Cria um banco de dados vazio chamado **postgres**

NOTE

- Anote a senha que será gerada para você se não for fornecida. Se você esquecer a senha, precisará redefiní-la usando o comando `az postgres flexible-server update`
- Se você não estiver usando o Ambiente do Serviço de Aplicativo, precisará habilitar Permitir acesso de qualquer IP do Azure usando esse comando.

```
az postgres flexible-server firewall-rule list --resource-group myresourcegroup --server-name mydemoserver --start-ip-address 0.0.0.0 --end-ip-address 0.0.0.0
```

Criar uma sub-rede para o ponto de extremidade do Serviço de Aplicativo

Agora, precisamos ter uma sub-rede delegada ao ponto de extremidade do Aplicativo Web do Serviço de Aplicativo. Execute o comando a seguir para criar uma sub-rede na mesma rede virtual do servidor de banco de dados criado.

```
az network vnet subnet create -g myresourcegroup --vnet-name VNETName --name webappsubnetName --address-prefixes 10.0.1.0/24 --delegations Microsoft.Web/serverFarms --service-endpoints Microsoft.Web
```

Anote os nomes da rede virtual e da sub-rede após esse comando, pois você precisará deles para adicionar a regra de Integração VNET ao aplicativo Web depois que ele for criado.

Criar um aplicativo Web

Nesta seção, você criará o host do aplicativo do Serviço de Aplicativo, conectará esse aplicativo ao banco de dados Postgres e implantará o código nesse host. Verifique se você está na raiz do repositório do código do aplicativo no terminal. Observação: o plano Básico não dá suporte à Integração VNET. Use o plano Standard ou Premium.

Crie um aplicativo do Serviço de Aplicativo (o processo de host) com o comando `az webapp up`

```
az webapp up --resource-group myresourcegroup --location westus2 --plan testappserviceplan --sku P2V2 --name mywebapp
```

NOTE

- Para o argumento `--location`, use a mesma localização usada para o banco de dados na seção anterior.
- Substitua por um nome exclusivo em todo o Azure (o ponto de extremidade do servidor é <https://<app-name>.azurewebsites.net>). Os caracteres permitidos para são A-Z, 0-9 e -. Um bom padrão é usar uma combinação do nome da empresa e um identificador de aplicativo.

Esse comando executa as seguintes ações, que podem levar alguns minutos:

- Crie o grupo de recursos se ele ainda não existir. (Neste comando, você usa o mesmo grupo de recursos no qual você criou o banco de dados anteriormente.)
- Criar o aplicativo do Serviço de Aplicativo se ele não existir.
- Habilitar o log padrão do aplicativo, se ainda não estiver habilitado.
- Carregar o repositório usando a implantação ZIP com a automação do build habilitada.

Adicionar o aplicativo Web à rede virtual

Use o comando `az webapp vnet-integration` para adicionar uma integração de rede virtual regional a um webapp. Substitua e pelo da rede virtual e da sub-rede que o servidor flexível está usando.

```
az webapp vnet-integration add -g myresourcegroup -n mywebapp --vnet VNETName --subnet webappsubnetName
```

Configurar as variáveis de ambiente para conexão com o banco de dados

Com o código implantado no Serviço de Aplicativo, a próxima etapa é conectar o aplicativo ao servidor flexível no Azure. O código do aplicativo espera encontrar informações sobre o banco de dados em diversas variáveis de ambiente. Para definir variáveis de ambiente no Serviço de Aplicativo, você cria "configurações do aplicativo" com o comando `az webapp config appsettings` set.

```
az webapp config appsettings set --settings DBHOST=<postgres-server-name>.postgres.database.azure.com"
DBNAME="postgres" DBUSER=<username>" DBPASS=<password>"
```

- Substitua `postgres-server-name`, `username`, `password` para o comando de servidor flexível que acaba de ser criado.
- Substitua e pelas credenciais que o comando também gerou para você.
- O grupo de recursos e o nome do aplicativo são extraídos dos valores armazenados em cache no arquivo `.azure/config`.
- O comando cria configurações chamadas `DBHOST`, `DBNAME`, `DBUSER` e `DBPASS`. Se o código do aplicativo estiver usando um nome diferente para as informações do banco de dados, use esses nomes para as configurações do aplicativo conforme mencionado no código.

Limpar os recursos

Limpe todos os recursos que você criou no tutorial usando o comando a seguir. Esse comando exclui todos os recursos nesse grupo de recursos.

```
az group delete -n myresourcegroup
```

Próximas etapas

[Mapear um nome DNS personalizado existente para o Serviço de Aplicativo do Azure](#)

Tutorial: Implantar o aplicativo Django com o Serviço de Aplicativo e o Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão prévia)

21/05/2021 • 10 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Neste tutorial, você aprenderá a implantar um aplicativo Django no Azure usando os Serviços de Aplicativos e o Banco de Dados do Azure para PostgreSQL – Servidor Flexível em uma rede virtual.

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Este artigo exige que você esteja executando a CLI do Azure versão 2.0 ou posterior localmente. Para ver a versão instalada, execute o comando `az --version`. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Você precisará fazer logon em sua conta usando o comando `login az`. Observe a propriedade **id** da saída do comando para o nome da assinatura correspondente.

```
az login
```

Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Selecione a ID da assinatura específica em sua conta usando o comando `az account set`. Substitua a propriedade **ID da assinatura** da saída `az login` por sua assinatura no espaço reservado da ID de assinatura.

```
az account set --subscription <subscription id>
```

Clonar ou baixar o aplicativo de exemplo

- [Clone do Git](#)
- [Download](#)

Clone o repositório de exemplo:

```
git clone https://github.com/Azure-Samples/djangoapp
```

Em seguida, acesse esta pasta:

```
cd djangoapp
```

O exemplo djangoapp contém o aplicativo de enquetes do Django controlado por dados que você obtém

segundo [Escrever seu primeiro aplicativo Django](#) na documentação do Django. O aplicativo concluído é fornecido aqui para fins de conveniência.

O exemplo também é modificado para ser executado em um ambiente de produção, como o Serviço de Aplicativo:

- As configurações de produção estão no arquivo `azuresite/production.py`. Os detalhes de desenvolvimento estão em `azuresite/settings.py`.
- O aplicativo usa as configurações de produção quando a variável de ambiente `DJANGO_ENV` é definida como "produção". Você criará essa variável de ambiente posteriormente no tutorial junto com outras usadas para a configuração do banco de dados PostgreSQL.

Essas alterações são específicas para configurar o Django para execução em qualquer ambiente de produção, e não são específicas para o Serviço de Aplicativo. Para saber mais, confira a [Lista de verificação de implantação do Django](#).

Criar um Servidor Flexível PostgreSQL em uma nova rede virtual

Crie um servidor flexível privado e um banco de dados dentro de uma VNET (rede virtual) usando o seguinte comando:

```
# Create Flexible server in a VNET  
  
az postgres flexible-server create --resource-group myresourcegroup --location westus2
```

Esse comando executa as seguintes ações, que podem levar alguns minutos:

- Crie o grupo de recursos se ele ainda não existir.
- Gera um nome do servidor, caso não tenha sido fornecido.
- Crie uma rede virtual para seu novo servidor PostgreSQL. **Anote o nome da rede virtual e o nome da sub-rede** criados para o servidor, pois você precisa adicionar o aplicativo Web à mesma rede virtual.
- Cria o nome de usuário e senha do administrador para o servidor, caso não fornecidos. **Tome nota do nome de usuário e a senha** a usar na próxima etapa.
- Crie um banco de dados `postgres` que possa ser usado para desenvolvimento. Você pode executar [psql para se conectar ao banco de dados](#) para criar um banco de dados diferente.

NOTE

Anote a senha que será gerada para você se não for fornecida. Se você esquecer a senha, precisará redefiní-la usando o comando `az postgres flexible-server update`

Implantar o código no Serviço de Aplicativo do Azure

Nesta seção, você criará o host do aplicativo do Serviço de Aplicativo, conectará esse aplicativo ao banco de dados Postgres e implantará o código nesse host.

Criar o aplicativo Web do Serviço de Aplicativo em uma rede virtual

No terminal, verifique se você está na raiz do repositório (`djangoapp`) que contém o código do aplicativo.

Crie um aplicativo do Serviço de Aplicativo (o processo de host) com o comando `az webapp up`:

```

# Create a web app
az webapp up --resource-group myresourcegroup --location westus2 --plan DjangoPostgres-tutorial-plan --sku
B1 --name <app-name>

# Enable VNET integration for web app.
# Replace <vnet-name> and <subnet-name> with the virtual network and subnet name that the flexible server is
using.

az webapp vnet-integration add -g myresourcegroup -n mywebapp --vnet <vnet-name> --subnet <subnet-name>

# Configure database information as environment variables
# Use the postgres server name , database name , username , password for the database created in the
previous steps

az webapp config appsettings set --settings DJANGO_ENV="production" DBHOST=".postgres.database.azure.com" DBNAME="postgres" DBUSER="" DBPASS=""
```

- Para o argumento `--location`, use a mesma localização usado para o banco de dados na seção anterior.
- Substitua `<app-name>` por um nome exclusivo em todo o Azure (o ponto de extremidade do servidor é `https://<app-name>.azurewebsites.net`). Os caracteres permitidos para `<app-name>` são `A - Z`, `0 - 9` e `-`. Um bom padrão é usar uma combinação do nome da empresa e um identificador de aplicativo.
- Criar o [plano do Serviço de Aplicativo](#) `DjangoPostgres-tutorial-plan` no tipo de preço Básico (B1) se ele não existir. `--plan` e `--sku` são opcionais.
- Criar o aplicativo do Serviço de Aplicativo se ele não existir.
- Habilitar o log padrão do aplicativo, se ainda não estiver habilitado.
- Carregar o repositório usando a implantação ZIP com a automação do build habilitada.
- O comando `az webapp vnet-integration` adiciona o aplicativo Web na mesma rede virtual que o servidor postgres.
- O código do aplicativo espera encontrar informações sobre o banco de dados em diversas variáveis de ambiente. Para definir as variáveis de ambiente no Serviço de Aplicativo, crie "configurações do aplicativo" usando o comando [az webapp config appsettings set](#).

TIP

Muitos comandos da CLI do Azure armazenam em cache parâmetros comuns, como o nome do grupo de recursos e o plano do Serviço de Aplicativo, no arquivo `.azure/config`. Como resultado, você não precisa especificar todos os mesmos parâmetros com comandos posteriores. Por exemplo, para reimplantar o aplicativo depois de fazer alterações, você pode simplesmente executar `az webapp up` novamente sem parâmetros.

Executar migrações de banco de dados do Django

As migrações de banco de dados do Django garantem que o esquema no PostgreSQL no banco de dados do Azure corresponda ao descrito em seu código.

1. Abra uma sessão SSH no navegador navegando até `https://<app-name>.scm.azurewebsites.net/webssh/host` e entre com suas credenciais de conta do Azure (não com as credenciais do servidor de banco de dados).
2. Na sessão SSH, execute os seguintes comandos (você pode colar os comandos usando **CTRL+Shift+V**):

```
cd site/wwwroot

# Activate default virtual environment in App Service container
source /antenv/bin/activate
# Install packages
pip install -r requirements.txt
# Run database migrations
python manage.py migrate
# Create the super user (follow prompts)
python manage.py createsuperuser
```

- O comando `createsuperuser` solicita suas credenciais de superusuário. Para este tutorial, use o nome de usuário padrão `root`, pressione **Enter** para o endereço de email ser deixado em branco e digite `postgres1` para a senha.

Criar uma pergunta de enquete no aplicativo

- Em um navegador, abra a URL `http://<app-name>.azurewebsites.net`. O aplicativo deve exibir a mensagem "Não há enquetes disponíveis" porque ainda não há enquetes específicas no banco de dados.
- Navegue até `http://<app-name>.azurewebsites.net/admin`. Entre usando as credenciais de superusuário da seção anterior (`root` e `postgres1`). Selecione **Enquetes**, selecione **Adicionar** ao lado de **Perguntas** e crie uma enquete com algumas opções.
- Navegue novamente para `http://<app-name>.azurewebsites.net/` para confirmar que as perguntas agora são apresentadas ao usuário. Responda às perguntas como desejar para gerar dados no banco de dados.

Parabéns! Você está executando um aplicativo Web Django, escrito em Python, no Serviço de Aplicativo do Azure para Linux, com um banco de dados Postgres ativo.

NOTE

O Serviço de Aplicativo detecta um projeto Django procurando um arquivo `wsgi.py` em cada subpasta, que o `manage.py startproject` cria por padrão. Quando o Serviço de Aplicativo encontra o arquivo, ele carrega o aplicativo Web Django. Para obter mais informações, confira [Configurar imagem interna do Python](#).

Fazer alterações no código e reimplantar

Nesta seção, você faz alterações locais no aplicativo e reimplanta o código no Serviço de Aplicativo. No processo, você configura um ambiente virtual do Python que dá suporte ao trabalho em andamento.

Executar o aplicativo localmente

Em uma janela de terminal, execute os comandos a seguir. Siga os prompts ao criar o superusuário:

```
# Configure the Python virtual environment
python3 -m venv venv
source venv/bin/activate

# Install packages
pip install -r requirements.txt
# Run Django migrations
python manage.py migrate
# Create Django superuser (follow prompts)
python manage.py createsuperuser
# Run the dev server
python manage.py runserver
```

Após o aplicativo Web ser totalmente carregado, o servidor de desenvolvimento do Django fornecerá a URL do

aplicativo local na mensagem "Iniciando o servidor de desenvolvimento em <http://127.0.0.1:8000/>. Feche o servidor com CTRL-BREAK".

```
Performing system checks...

System check identified no issues (0 silenced).
June 24, 2020 - 10:27:14
Django version 2.2.13, using settings 'azuresite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Teste o aplicativo localmente com as seguintes etapas:

1. Navegue para <http://localhost:8000> em um navegador, o que deverá exibir a mensagem "Não há enquetes disponíveis".
2. Navegue até <http://localhost:8000/admin> e entre usando o usuário administrador que você criou anteriormente. Em **Enquetes**, selecione **Adicionar** novamente ao lado de **Perguntas** e crie uma enquete com algumas opções.
3. Vá para <http://localhost:8000> novamente e responda à pergunta para testar o aplicativo.
4. Pare o servidor Django pressionando **CTRL+C**.

Ao ser executado localmente, o aplicativo está usando um banco de dados SQLite3 local e não interfere no banco de dados de produção. Você também poderá usar um banco de dados PostgreSQL local, se desejar, para simular melhor seu ambiente de produção.

Atualizar o aplicativo

Em `polls/models.py`, localize a linha que começa com `choice_text` e altere o parâmetro `max_length` para 100:

```
# Find this lie of code and set max_length to 100 instead of 200
choice_text = models.CharField(max_length=100)
```

Como você alterou o modelo de dados, crie uma migração do Django e migre o banco de dados:

```
python manage.py makemigrations
python manage.py migrate
```

Execute o servidor de desenvolvimento novamente com `python manage.py runserver` e teste o aplicativo em <http://localhost:8000/admin>:

Pare o servidor Web Django novamente com **CTRL+C**.

Reimplantar o código no Azure

Execute o seguinte comando na raiz do repositório:

```
az webapp up
```

Esse comando usa os parâmetros armazenados em cache no arquivo `.azure/config`. Como o Serviço de Aplicativo detecta que o aplicativo já existe, ele apenas reimplanta o código.

Executar migrações novamente no Azure

Como você fez alterações no modelo de dados, é necessário executar novamente as migrações de banco de dados no Serviço de Aplicativo.

Abra uma sessão SSH novamente no navegador acessando <https://<app-name>.azurewebsites.net>

`name>.scm.azurewebsites.net/webssh/host`. Em seguida, execute os comandos a seguir:

```
cd site/wwwroot

# Activate default virtual environment in App Service container
source /antenv/bin/activate
# Run database migrations
python manage.py migrate
```

Examinar o aplicativo na produção

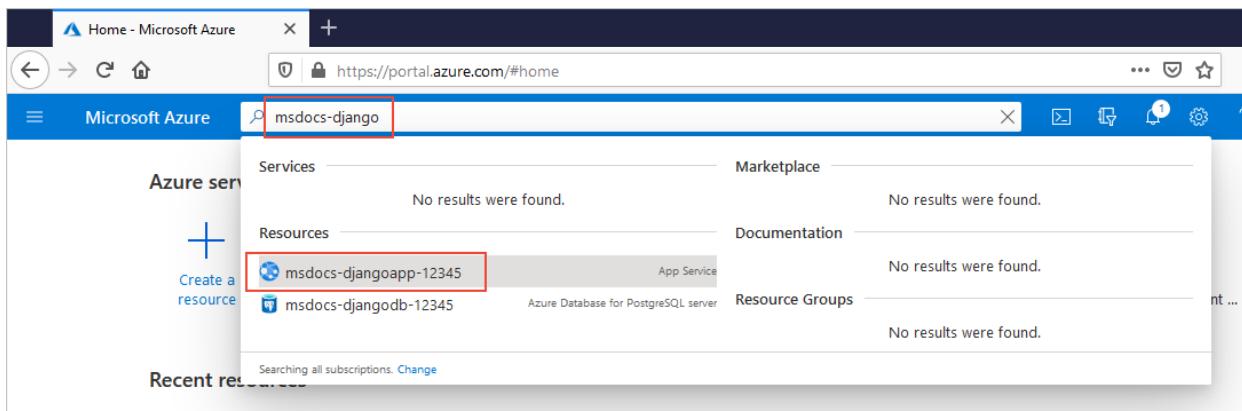
Navegue até `http://<app-name>.azurewebsites.net` e teste o aplicativo novamente em produção. (Como você alterou apenas o comprimento de um campo do banco de dados, a alteração só será perceptível se você tentar inserir uma resposta mais longa ao criar uma pergunta.)

TIP

Você pode usar [django-storages](#) para armazenar ativos de mídia e estáticos no armazenamento do Azure. Você pode usar a CDN do Azure para gzipping para arquivos estáticos.

Gerenciar seu aplicativo no portal do Azure

No [portal do Azure](#), pesquise pelo nome do aplicativo e selecione-o nos resultados.



The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar with the text "msdocs-django". Below the search bar, the main content area has sections for "Services" and "Marketplace", both of which show "No results were found.". Under the "Resources" section, there are two items listed: "msdocs-djangoapp-12345" and "msdocs-djangoapp-12345". The first item is highlighted with a red box. To the right of the resources, there are sections for "App Service", "Documentation", and "Resource Groups", all of which show "No results were found.". At the bottom left, there's a sidebar with a "Create a resource" button and a "Recent resources" section.

Por padrão, o portal mostra a página de **Visão geral** do aplicativo, que fornece uma exibição do desempenho geral. Aqui você também pode executar tarefas básicas de gerenciamento como procurar, parar, reiniciar e excluir. As guias no lado esquerdo da página mostram as páginas de configuração diferentes que você pode abrir.

The screenshot shows the Microsoft Azure portal interface for managing an App Service. The main page displays the app's name, 'msdocs-djangoapp-12345', and its status as 'Running' in 'West US 2'. Key details include the Resource group ('DjangoPostgres-tutorial-rg'), App Service Plan ('DjangoPostgres-tutorial-plan (B1: 1)'), and various URLs and hostnames. On the left, a sidebar provides navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Events (preview), Deployment (Quickstart, Deployment slots, Deployment Center), and Settings (Configuration, Authentication / Authorization, Application Insights, Identity). The central area features two cards: 'Diagnose and solve problems' and 'App Service Advisor', followed by three performance charts: 'Http 5xx' (1 occurrence at 0.8 kB), 'Data In' (180kB, 160kB, 140kB, 120kB), and 'Data Out' (800kB, 700kB, 600kB).

Limpar os recursos

Se você quiser manter o aplicativo ou prosseguir para o próximo tutorial, pule para as [Próximas etapas](#). Caso contrário, para evitar incorrer em encargos contínuos, você pode excluir o grupo de recursos criado para este tutorial:

```
az group delete -g myresourcegroup
```

O comando usa o nome do grupo de recursos armazenado em cache no arquivo `.azure/config`. Ao excluir o grupo de recursos, você também desaloca e exclui todos os recursos contidos nele.

Próximas etapas

Saiba como mapear um nome DNS personalizado para seu aplicativo:

[Tutorial: Mapear o nome DNS personalizado para seu aplicativo](#)

Saiba como o Serviço de Aplicativo executa um aplicativo Python:

[Configurar o aplicativo Python](#)

Servidores – Banco de Dados do Azure para PostgreSQL – Servidor Flexível

09/08/2021 • 3 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo apresenta diretrizes e considerações para trabalhar com o Banco de Dados do Azure para PostgreSQL – Servidor Flexível.

O que é um servidor do Banco de Dados do Azure para PostgreSQL?

Um servidor na opção de implantação do Banco de Dados do Azure para PostgreSQL – Servidor Flexível é um ponto administrativo central para vários bancos de dados. É a mesma construção de servidor PostgreSQL com a qual talvez você já esteja familiarizado no mundo local. Especificamente, o serviço PostgreSQL é gerenciado, oferece garantias de desempenho, expõe acesso e recursos no nível do servidor.

Um Banco de Dados do Azure para servidor PostgreSQL:

- É criado dentro de uma assinatura do Azure.
- É o recurso pai para bancos de dados.
- Fornece um namespace para bancos de dados.
- É um contêiner com semântica de tempo de vida fortes – exclua um servidor e ele excluirá os bancos de dados contidos.
- Coloca recursos em uma região.
- Fornece um ponto de extremidade de conexão para acesso ao servidor e ao banco de dados
- Fornece o escopo para políticas de gerenciamento que se aplicam a seus bancos de dados: logons, firewall, usuários, funções, configurações etc.
- Está disponível em várias versões. Para saber mais, confira [Versões do banco de dados PostgreSQL com suporte](#).
- É extensível pelos usuários. Para saber mais, confira [Extensões do PostgreSQL](#).

Dentro de um banco de dados do Azure para o servidor PostgreSQL, você pode criar um ou mais bancos de dados. Você pode optar por criar um banco de dados por servidor para utilizar todos os recursos ou criar vários bancos de dados para compartilhar os recursos. Os preços são estruturados por servidor, com base na configuração do tipo de preço, vCores e armazenamento (GB). Para obter mais informações, veja [Opções de computação e armazenamento](#).

Como se conectar e autenticar no servidor de banco de dados?

Os elementos a seguir ajudam a garantir o acesso seguro ao seu banco de dados:

CONCEITO DE SEGURANÇA

DESCRIÇÃO

CONCEITO DE SEGURANÇA	DESCRIÇÃO
Autenticação e autorização	O Banco de Dados do Azure para servidor PostgreSQL oferece suporte à autenticação de PostgreSQL nativa. Você pode se conectar e autenticar no servidor com logon de administrador do servidor.
Protocolo	O serviço oferece suporte a um protocolo baseado em mensagem usado pelo PostgreSQL.
TCP/IP	O protocolo tem suporte em TCP/IP e em soquetes de domínio do Unix.
Firewall	Para ajudar a proteger seus dados, uma regra de firewall impede todo acesso ao servidor e seus bancos de dados até que você especifique quais computadores têm permissão. Confira Regras de firewall do Banco de Dados do Azure para servidor PostgreSQL .

Gerenciando o servidor

Você pode gerenciar o Banco de Dados do Azure para servidores PostgreSQL usando o [Portal do Azure](#) ou a [CLI do Azure](#).

Ao criar um servidor, você configura as credenciais de seu usuário administrador. O usuário administrador é o usuário com privilégio mais elevado no servidor. Ele pertence à função `azure_pg_admin`. Essa função não tem permissões completas de superusuário.

O atributo de superusuário do PostgreSQL é atribuído a `azure_superuser`, que pertence ao serviço gerenciado. Você não tem acesso a essa função.

Um Banco de Dados do Azure para PostgreSQL possui bancos de dados padrão:

- `postgres` – um banco de dados padrão a que você poderá se conectar após seu servidor ser criado.
- `azure_maintenance` – este banco de dados é usado para separar os processos que fornecem o serviço gerenciado das ações do usuário. Você não tem acesso a esse banco de dados.

Parâmetros do Servidor

Os parâmetros de servidor PostgreSQL determinam a configuração do servidor. No Banco de Dados do Azure para PostgreSQL, a lista de parâmetros pode ser exibida e editada usando o portal do Azure ou a CLI do Azure.

Como um serviço gerenciado para Postgres, os parâmetros configuráveis no banco de dados do Azure para PostgreSQL são um subconjunto dos parâmetros em uma instância local do Postgres (para obter mais informações sobre parâmetros Postgres, consulte o [PostgreSQL documentação](#)). O banco de dados do Azure para servidor PostgreSQL está habilitado com valores padrão para cada parâmetro na criação. Alguns parâmetros que necessitariam de um reinício de servidor ou de acesso de superusuário para as mudanças terem efeito não podem ser configurados pelo usuário.

Próximas etapas

- Para obter uma visão geral do serviço, confira [Visão geral do Banco de Dados para PostgreSQL](#).
- Para obter informações sobre cotas de recursos e limitações específicas com base em sua [configuração](#), consulte [Opções de computação e armazenamento](#).
- Exibir e editar os parâmetros de servidor por meio de [portal do Azure](#) ou [CLI do Azure](#).

Versões principais do PostgreSQL com suporte no banco de dados do Azure para PostgreSQL – Servidor flexível

09/08/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Banco de Dados do Azure para PostgreSQL - Servidor flexível dá suporte às seguintes versões principais no momento:

PostgreSQL versão 13

A versão secundária atual é 13.3. Veja a [documentação do PostgreSQL](#) para saber mais sobre os aprimoramentos e as correções nesta versão. Novos servidores serão criados com essa versão secundária.

PostgreSQL versão 12

A versão secundária atual é 12.7. Veja a [documentação do PostgreSQL](#) para saber mais sobre os aprimoramentos e as correções nesta versão. Novos servidores serão criados com essa versão secundária. Os servidores existentes serão atualizados automaticamente para a versão secundária mais recente, e com suporte, em sua janela de manutenção agendada futura.

PostgreSQL versão 11

A versão secundária atual é 11.12. Veja a [documentação do PostgreSQL](#) para saber mais sobre os aprimoramentos e as correções nesta versão. Novos servidores serão criados com essa versão secundária. Os servidores existentes serão atualizados automaticamente para a versão secundária mais recente, e com suporte, em sua janela de manutenção agendada futura.

PostgreSQL versão 10 e anteriores

Não há suporte para PostgreSQL versão 10 e anteriores para o Banco de Dados do Azure para PostgreSQL – Servidor flexível. Use a opção de implantação de [Servidor único](#) se você precisar de versões mais antigas.

Como gerenciar atualizações

O projeto PostgreSQL emite regularmente versões secundárias para corrigir bugs relatados. O Banco de Dados do Azure para PostgreSQL corrige automaticamente os servidores com versões secundárias durante as implantações mensais do serviço.

Ainda não há suporte para a automação da atualização de versão principal. Por exemplo, não há nenhuma atualização automática do PostgreSQL 11 para PostgreSQL 12.

Opções de Processamento e Armazenamento - Banco de Dados do Azure para PostgreSQL - Servidor Flexível

21/05/2021 • 9 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

É possível criar um servidor do Banco de Dados do Azure para PostgreSQL em um dos três tipos de preço diferentes: Com Capacidade de Intermitência, Uso Geral e Otimizado para Memória. Os tipos de preço são diferenciados pela quantidade de computação nos vCores que pode ser provisionada, pela memória por vCore e pela tecnologia de armazenamento usada para armazenar os dados. Todos os recursos são provisionados no nível do servidor PostgreSQL. Um servidor pode ter um ou vários bancos de dados.

RECURSO / CAMADA	COM CAPACIDADE DE INTERMITÊNCIA	USO GERAL	OTIMIZADO PARA MEMÓRIA
vCores	1, 2	2, 4, 8, 16, 32, 48, 64	2, 4, 8, 16, 32, 48, 64
Memória por vCore	Variável	4 GB	6.75 a 8 GB
Tamanho de armazenamento	32 GB a 16 TB	32 GB a 16 TB	32 GB a 16 TB
Período de retenção do backup de banco de dados	7 a 35 dias	7 a 35 dias	7 a 35 dias

Para escolher um tipo de preço, use a tabela a seguir como ponto de partida.

TIPO DE PREÇO	CARGAS DE TRABALHO DE DESTINO
Com capacidade de intermitência	Ideal para cargas de trabalho que não precisam de toda a CPU continuamente.
Uso Geral	A maioria das cargas de trabalho que exigem a computação e a memória balanceadas com a taxa de transferência de E/S escalonável. Os exemplos incluem servidores para hospedar aplicativos Web e móveis e outros aplicativos empresariais.
Otimizado para memória	Cargas de trabalho de banco de dados de alto desempenho que exigem desempenho na memória para o processamento de transações mais rápido e com simultaneidade mais alta. Os exemplos incluem servidores para o processamento de dados em tempo real e aplicativos analíticos ou transacionais de alto desempenho.

Depois de criar um servidor, a camada de computação, o número de vCores e o tamanho do armazenamento podem ser aumentados ou reduzidos em segundos. Você também pode ajustar de forma independente o

período de retenção de backup para cima ou para baixo. Para obter mais informações, consulte a seção [Recursos de dimensionamento](#).

Camadas de computação, vCores e tipos de servidor

Os recursos de processamento podem ser selecionados com base na camada e no tamanho. vCores representam a CPU lógica do hardware subjacente.

As especificações detalhadas dos tipos de servidor disponíveis são as seguintes:

NOME DO SKU	VCORES	TAMANHO DA MEMÓRIA	MÁXIMO DE IOPS COM SUPORTE	MÁXIMO DE LARGURA DE BANDA DE E/S COM SUPORTE
Com capacidade de intermitência				
B1ms	1	2 GiB	640	15 MiB/segundo
B2s	2	4 GiB	1280	15 MiB/segundo
Uso Geral				
D2s_v3	2	8 GiB	3200	48 MiB/segundo
D4s_v3	4	16 GiB	6400	96 MiB/segundo
D8s_v3	8	32 GiB	12800	192 MiB/segundo
D16s_v3	16	64 GiB	18000	384 MiB/segundo
D32s_v3	32	128 GiB	18000	750 MiB/segundo
D48s_v3	48	192 GiB	18000	750 MiB/segundo
D64s_v3	64	256 GiB	18000	750 MiB/segundo
Otimizado para memória				
E2s_v3	2	16 GiB	3200	48 MiB/segundo
E4s_v3	4	32 GiB	6400	96 MiB/segundo
E8s_v3	8	64 GiB	12800	192 MiB/segundo
E16s_v3	16	128 GiB	18000	384 MiB/segundo
E32s_v3	32	256 GiB	18000	750 MiB/segundo
E48s_v3	48	384 GiB	18000	750 MiB/segundo
E64s_v3	64	432 GiB	18000	750 MiB/segundo

Armazenamento

O armazenamento provisionado é a quantidade de capacidade de armazenamento disponível para o Banco de Dados do Azure para servidor PostgreSQL. O armazenamento é usado para os arquivos de banco de dados, os logs de transações e os logs do servidor PostgreSQL. A quantidade total de armazenamento que você provisiona também define a capacidade disponível para o servidor.

O armazenamento está disponível nos seguintes tamanhos fixos:

TAMANHO DO DISCO	IOPS
32 GiB	Provisionado 120, até 3.500
64 GiB	Provisionado 240, até 3.500
128 GiB	Provisionado 500, até 3.500
256 GiB	Provisionado 1100, até 3.500
512 GiB	Provisionado 2300, até 3.500
1 TiB	5.000
2 TiB	7.500
4 TiB	7.500
8 TiB	16.000
16 TiB	18.000

Observe que os IOPS também são restritos pelo tipo de VM. Embora você possa selecionar qualquer tamanho de armazenamento independentemente do tipo de servidor, talvez não seja possível usar todos os IOPS fornecidos pelo armazenamento, especialmente quando você escolhe um servidor com um pequeno número de vCores.

Você pode adicionar mais capacidade de armazenamento durante e após a criação do servidor.

NOTE

O armazenamento só pode ser escalado verticalmente, não horizontalmente.

Você pode monitorar o consumo de E/S no Portal do Azure ou usando os comandos da CLI do Azure. As métricas relevantes para monitorar são o [limite de armazenamento](#), [porcentagem de armazenamento](#), [armazenamento usado](#) e [porcentagem de E/S](#).

IOPS máximo para sua configuração

NOME DO SKU	TAMANHO DE ARMAZENAMENTO EM GIB	32	64	128	256	512	1.024	2.048	4.096	8.192	16.384
	IOPS máximo	120	240	500	1100	2300	5.000	7500	7500	16000	18000
Com capacidade de intermitência											
B1ms	640 IOPS	120	240	500	640*	640*	640*	640*	640*	640*	640*
B2s	1280 IOPS	120	240	500	1100	1280*	1280*	1280*	1280*	1280*	1280*
Uso Geral											
D2s_v3	3200 IOPS	120	240	500	1100	2300	3200*	3200*	3200*	3200*	3200*
D4s_v3	6.400 IOPS	120	240	500	1100	2300	5.000	6400*	6400*	6400*	6400*
D8s_v3	12800 IOPS	120	240	500	1100	2300	5.000	7500	7500	12800*	12800*
D16s_v3	18.000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000
D32s_v3	18.000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000
D48s_v3	18.000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000
D64s_v3	18.000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000

NOME DO SKU	TAMANHO DE ARMAZENAMENTO OEM	GIB	32	64	128	256	512	1.024	2.048	4.096	8.192	16.384
Otimizado para memória												
E2s_v3	3200 IOPS	120	240	500	1100	2300	3200*	3200*	3200*	3200*	3200*	3200*
E4s_v3	6.400 IOPS	120	240	500	1100	2300	5.000	6400*	6400*	6400*	6400*	6400*
E8s_v3	12800 IOPS	120	240	500	1100	2300	5.000	7500	7500	12800*	12800*	12800*
E16s_v3	18000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000	18000
E32s_v3	18000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000	18000
E48s_v3	18000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000	18000
E64s_v3	18000 IOPS	120	240	500	1100	2300	5.000	7500	7500	16000	18000	18000

Quando marcados com um *, os IOPS são limitados pelo tipo de VM selecionado. Caso contrário, os IOPS serão limitados pelo tamanho de armazenamento selecionado.

NOTE

Você pode ver IOPS mais altos nas métricas devido ao estouro no nível do disco. Consulte a [documentação](#) para obter mais detalhes.

Largura de banda máxima de E/S (MiB/s) para sua configuração

NOME DO SKU	TAMANHO DE ARMAZENAMENTO, GIB	32	64	128	256	512	1.024	2.048	4.096	8.192	16.384
	Largura de banda de armazenamento, MiB/segundo	25	50	100	125	150	200	250	250	500	750
	Com capacidade de intermitência										
B1ms	10 MiB/s egundo	10*	10*	10*	10*	10*	10*	10*	10*	10*	10*
B2s	15 MiB/s egundo	15*	15*	15*	15*	15*	15*	15*	15*	15*	15*
Uso Geral											
D2s_v3	48 MiB/s egundo	25	48*	48*	48*	48*	48*	48*	48*	48*	48*
D4s_v3	96 MiB/s egundo	25	50	96*	96*	96*	96*	96*	96*	96*	96*
D8s_v3	192 MiB/s egundo	25	50	100	125	150	192*	192*	192*	192*	192*
D16s_v3	384 MiB/s egundo	25	50	100	125	150	200	250	250	384*	384*

NOME DO SKU	TAMANHO DE ARMAZENAMENTO, GIB	32	64	128	256	512	1.024	2.048	4.096	8.192	16.384
D32s_v3	750 MiB/s egundo	25	50	100	125	150	200	250	250	500	750
D48s_v3	750 MiB/s egundo	25	50	100	125	150	200	250	250	500	750
D64s_v3	750 MiB/s egundo	25	50	100	125	150	200	250	250	500	750
Otimizado para memória											
E2s_v3	48 MiB/s egundo	25	48*	48*	48*	48*	48*	48*	48*	48*	48*
E4s_v3	96 MiB/s egundo	25	50	96*	96*	96*	96*	96*	96*	96*	96*
E8s_v3	192 MiB/s egundo	25	50	100	125	150	192*	192*	192*	192*	192*
E16s_v3	384 MiB/s egundo	25	50	100	125	150	200	250	250	384*	384*
E32s_v3	750 MiB/s egundo	25	50	100	125	150	200	250	250	500	750
E48s_v3	750 MiB/s egundo	25	50	100	125	150	200	250	250	500	750

NOME DO SKU	TAMANHO DE ARMAZENAMENTO, GiB	32	64	128	256	512	1.024	2.048	4.096	8.192	16.384
E64s_v3	750 MiB/s segundo	25	50	100	125	150	200	250	250	500	750

Quando marcado com uma *, a largura de banda de E/S é limitada pelo tipo de VM selecionado. Caso contrário, a largura de banda de E/S é limitada pelo tamanho de armazenamento selecionado.

Alcançando o limite de armazenamento

Quando você atingir o limite de armazenamento, o servidor começará a retornar erros e evitará outras modificações. Isso também pode causar problemas com outras atividades operacionais, como backups e arquivamento WAL.

Para evitar essa situação, quando o uso do armazenamento atinge 95% ou se a capacidade disponível é inferior a 5 GiB, o servidor é alternado automaticamente para o **modo somente leitura**.

É recomendável monitorar ativamente o espaço em disco que está em uso e aumentar o tamanho do disco antes de qualquer situação de armazenamento insuficiente. Você pode configurar um alerta para notificá-lo quando o armazenamento do servidor estiver se aproximando do disco para que você possa evitar problemas com disco insuficiente. Para mais informações, consulte a documentação em [como configurar um alerta](#).

Aumento automático de armazenamento

O aumento automático de armazenamento ainda não está disponível para Servidor Flexível.

Backup

O serviço faz backups do servidor automaticamente. É possível definir um período de retenção num intervalo de 7 a 35 dias. Saiba mais sobre backups no [artigo sobre conceitos](#).

Escalar recursos

Após criar o servidor, você poderá, independentemente, alterar vCores, a quantidade de armazenamento e o período de retenção de backup. O número de vCores pode ser dimensionado para cima ou para baixo. Os vCores e o período de retenção de backup podem ser aumentados ou reduzidos de 7 a 35 dias. O tamanho de armazenamento só pode ser aumentado. O dimensionamento dos recursos pode ser feito por meio do portal ou da CLI do Azure.

NOTE

O tamanho de armazenamento só pode ser aumentado. Você não poderá voltar para um tamanho de armazenamento menor após o aumento.

Quando você altera a camada de processamento ou o número de vCores, o servidor é reiniciado para que o novo tipo de servidor entre em vigor. Durante um momento enquanto o sistema muda para o novo servidor, nenhuma nova conexão pode ser estabelecida e todas as transações não confirmadas são revertidas. Esse período varia, mas na maioria dos casos fica abaixo um minuto. O dimensionamento do armazenamento funciona da mesma maneira e também requer uma breve reinicialização.

Alterar o período de retenção de backup é uma operação online.

Precos

Para as informações mais recentes sobre preços, consulte a [página de preços](#) do serviço. Para ver os custos da configuração desejada, o [Portal do Azure](#) mostra o custo mensal na guia **Tipo de preço** com base nas opções que você seleciona. Se você não tiver uma assinatura do Azure, poderá usar a calculadora de preços do Azure para obter um preço estimado. No site da [Calculadora de preços do Azure](#), selecione **Adicionar itens**, expanda a categoria **Bancos de dados** e escolha **Banco de Dados do Azure para PostgreSQL** para personalizar as opções.

Próximas etapas

- Saiba como [criar um servidor PostgreSQL no portal](#).
- Conheça os [limites de serviço](#).

Visão geral da rede – Banco de Dados do Azure para PostgreSQL – Servidor flexível

09/08/2021 • 9 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo descreve os conceitos de conectividade e rede para o Banco de Dados do Azure para PostgreSQL - Servidor flexível.

Escolher uma opção de rede

Você tem duas opções de rede para o Banco de Dados do Azure para PostgreSQL - Servidor flexível. As opções são **acesso privado (integração de VNet)** e **acesso público (endereços IP permitidos)**. Na criação do servidor, você deve escolher uma opção.

NOTE

A opção de rede não pode ser alterada depois que o servidor for criado.

- **Acesso privado (integração de VNet)** – Você pode implantar seu servidor flexível em sua [Rede Virtual do Azure](#). As redes virtuais do Azure fornecem comunicação de rede privada e segura. Os recursos em uma rede virtual podem se comunicar por meio de endereços IP privados.

Escolha a opção de Integração de VNet se desejar as seguintes funcionalidades:

- Conectar-se de recursos do Azure na mesma rede virtual com seu servidor flexível usando endereços IP privados
- Usar a VPN ou o ExpressRoute para se conectar de recursos que não são do Azure com seu servidor flexível
- O servidor flexível tem um ponto de extremidade público

- **Acesso público (endereços IP permitidos)** – O servidor flexível é acessado por meio de um ponto de extremidade público. O ponto de extremidade público é um endereço DNS que poderia ser resolvido publicamente. A frase "endereços IP permitidos" refere-se a um intervalo de IPs que você escolhe para conceder permissão para acessar o servidor. Essas permissões são chamadas **regras de firewall**.

Escolha o método de acesso público se desejar os seguintes recursos:

- Conectar-se a partir de recursos do Azure que não dão suporte a redes virtuais
- Conectar-se a partir de recursos fora de um Azure que não estão conectados por VPN ou ExpressRoute
- O servidor flexível tem um ponto de extremidade público

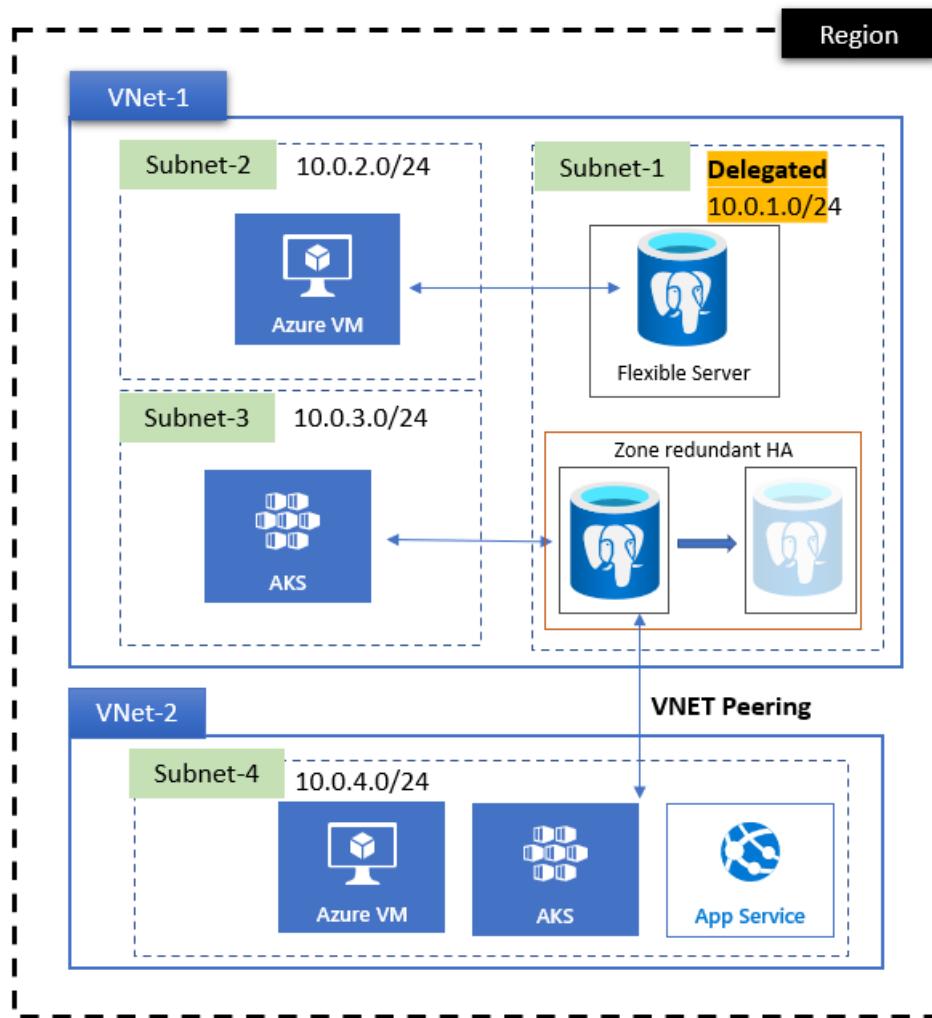
As seguintes características se aplicam se você escolher usar o acesso privado ou a opção de acesso público:

- As conexões de endereços IP permitidos precisam ser autenticadas no servidor PostgreSQL com credenciais válidas
- A [criptografia de conexão](#) está disponível para o tráfego de rede

- O servidor tem um FQDN (nome de domínio totalmente qualificado). Para a propriedade nome do host em cadeias de conexão, é recomendável usar o FQDN em vez de um endereço IP.
- As duas opções controlam o acesso no nível do servidor, não no nível do banco de dados ou da tabela. Você usaria as propriedades de funções do PostgreSQL para controlar o banco de dados, a tabela e o acesso a outros objetos.

Acesso privado (integração da VNet)

O acesso privado com integração de rede virtual (vnet) fornece comunicação privada e segura para o servidor flexível PostgreSQL.



No diagrama acima,

1. Os servidores flexíveis são injetados em uma sub-rede delegada – 10.0.1.0/24 da VNET VNet-1.
2. Os aplicativos implantados em sub-redes diferentes na mesma vnet podem acessar os servidores flexíveis diretamente.
3. Os aplicativos implantados em uma VNET VNet-2 diferente não têm acesso direto a servidores flexíveis. Você precisa executar o [emparelhamento VNET de zona DNS privada](#) antes que possa acessar o servidor flexível.

Conceitos de rede virtual

Estes são alguns conceitos que você deve conhecer ao usar redes virtuais com servidores flexíveis PostgreSQL.

- **Rede virtual** – uma VNet (rede virtual) do Azure contém um espaço de endereço IP privado configurado para o uso. Visite a [Visão geral da Rede Virtual do Azure](#) para saber mais sobre redes virtuais do Azure.

A rede virtual deve estar na mesma região do Azure que o servidor flexível.

- **Sub-rede delegada** – uma rede virtual contém sub-redes. As sub-redes permitem segmentar a rede virtual em espaços de endereço menores. Os recursos do Azure são implantados em sub-redes específicas dentro de uma rede virtual.

O servidor flexível PostgreSQL deve estar em uma sub-rede **delegada** somente para uso do servidor flexível PostgreSQL. Essa delegação significa que somente os Servidores Flexíveis do Banco de Dados do Azure para PostgreSQL podem usar essa sub-rede. Nenhum outro tipo de recurso do Azure pode estar na sub-rede delegada. Você delega uma sub-rede atribuindo sua propriedade de delegação como Microsoft.DBforPostgreSQL/flexibleServers.

IMPORTANT

Os nomes incluindo `AzureFirewallSubnet`, `AzureFirewallManagementSubnet`, `AzureBastionSubnet` e `GatewaySubnet` são nomes reservados no Azure. Não os use como sendo o nome da sub-rede.

- **NSG (grupos de segurança de rede)** - As regras de segurança em grupos de segurança de rede permitem filtrar o tipo de tráfego de rede que pode fluir para dentro e para fora das sub-redes da rede virtual e dos adaptadores de rede. Confira [Visão geral do Grupo de Segurança de Rede](#) para obter mais informações.
- **Integração de zona de DNS privado** – A integração de zona de DNS privado do Azure permite que você resolva o DNS privado na VNET atual ou qualquer VNET emparelhada na região em que a zona de DNS privado esteja vinculada. Consulte a [documentação da zona de DNS privado](#) para obter mais detalhes.

Saiba como criar um servidor flexível com acesso privado (integração VNet) [no portal do Azure](#) ou [na CLI do Azure](#).

Integração com o servidor DNS personalizado

Se estiver usando o servidor DNS personalizado, você deverá usar um encaminhador DNS para resolver o FQDN do Banco de Dados do Azure para PostgreSQL – Servidor Flexível. O endereço IP do encaminhador deve ser [168.63.129.16](#). O servidor DNS personalizado deve estar dentro da VNet ou acessível por meio da configuração do servidor DNS da VNET. Consulte a [resolução de nomes que usa o próprio servidor DNS](#) para saber mais.

Zona de DNS privado e emparelhamento VNET

As configurações de zona de DNS privado e emparelhamento VNET são independentes um do outro.

- Por padrão, uma nova zona de DNS privado é provisionada automaticamente por servidor usando o nome do servidor fornecido. No entanto, se você desejar configurar sua própria zona de DNS privado para usar com o servidor flexível, consulte a documentação de [visão geral de DNS privado](#).
- Se desejar se conectar ao servidor flexível de um cliente provisionado em outra VNET, será necessário vincular a zona de DNS privado à VNET. Confira [como vincular a documentação da rede virtual](#).

NOTE

Os nomes da zona de DNS privado que terminam com `private.postgres.database.azure.com` só podem ser vinculados.

Cenários de rede virtual sem suporte

- Ponto de extremidade público (ou IP ou DNS) – um servidor flexível implantado em uma rede virtual não pode ter um ponto de extremidade público
- Depois que o servidor flexível for implantado em uma rede virtual e sub-rede, você não poderá movê-lo

para outra rede virtual ou sub-rede. Não é possível mover a rede virtual para outro grupo de recursos ou assinatura.

- O tamanho da sub-rede (espaços de endereço) não pode ser aumentado após a existência de recursos na sub-rede
- Não há suporte para o emparelhamento de VNets entre regiões

Acesso público (endereços IP permitidos)

As características do método de acesso público incluem:

- Somente os endereços IP que você permitir terão permissão para acessar o servidor flexível PostgreSQL. Por padrão, nenhum endereço IP é permitido. Você pode adicionar endereços IP durante a criação do servidor ou depois.
- O servidor PostgreSQL tem um nome DNS que pode ser resolvido publicamente
- O servidor flexível não está em uma das redes virtuais do Azure
- O tráfego de rede de e para o servidor não passa por uma rede privada. O tráfego usa os caminhos gerais da Internet.

Regras de firewall

A concessão de permissão a um endereço IP é chamada de regra de firewall. Se uma tentativa de conexão vier de um endereço IP sem permissão, o cliente de origem verá um erro.

Saiba como criar um servidor flexível com acesso público (endereços IP permitidos) [no portal do Azure](#) ou [na CLI do Azure](#).

Permitir todos os endereços IP do Azure

Se um endereço IP de saída fixo não estiver disponível para o serviço do Azure, você poderá considerar habilitar conexões de todos os endereços IP do datacenter do Azure.

IMPORTANT

A opção **permitir acesso público dos serviços e recursos do Azure no Azure** configura o firewall para permitir todas as conexões do Azure, incluindo conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

Solução de problemas de acesso público

Considere os seguintes pontos quando o acesso ao Banco de Dados do Microsoft Azure para o serviço de servidor PostgreSQL não se comportar conforme o esperado:

- **As alterações à lista de permitidos ainda não entraram em vigor:** pode ocorrer um atraso de cinco minutos para que as alterações na configuração de firewall do Banco de Dados do Azure para PostgreSQL entrem em vigor.
- **Falha na autenticação:** se um usuário não tiver permissões no servidor do Banco de Dados do Azure para PostgreSQL ou se a senha usada estiver incorreta, a conexão com o servidor do Banco de Dados do Azure para PostgreSQL será negada. A criação de uma configuração de firewall apenas fornece aos clientes uma oportunidade de tentar se conectar ao servidor. O cliente ainda deve fornecer as credenciais de segurança necessárias.
- **Endereço IP dinâmico de cliente:** se você tiver uma conexão com a Internet com endereço IP dinâmico e estiver com dificuldades para atravessar o firewall, tente uma das seguintes soluções:
 - Peça ao seu Provedor de serviços de Internet (ISP) o intervalo de endereços IP atribuído aos computadores clientes que acessarão o servidor de Banco de Dados do Azure para servidor PostgreSQL e, em seguida, adicione o intervalo de endereços IP como uma regra de firewall.

- Obtenha o endereçamento IP estático para os computadores cliente e adicione os endereços IP estáticos como uma regra de firewall.
- **A regra de firewall não está disponível para o formato IPv6:** as regras de firewall devem estar no formato IPv4. Se você especificar regras de firewall no formato IPv6, ele mostrará o erro de validação.

Nome do host

Independentemente da opção de rede que você escolher, é recomendável usar sempre um FQDN (nome de domínio totalmente qualificado) como nome do host ao se conectar ao servidor flexível. Não há garantia de que o endereço IP do servidor permanecerá estático. Usar o FQDN ajudará você a evitar fazer alterações na cadeia de conexão.

Exemplo

- Recomendado `hostname = servername.postgres.database.azure.com`
- Sempre que possível, evite usar `hostname = 10.0.0.4` (um endereço privado) ou `hostname = 40.2.45.67` (um endereço público)

TLS e SSL

O Banco de Dados do Azure para PostgreSQL - Servidor Flexível dá suporte à conexão dos aplicativos clientes ao serviço do PostgreSQL usando TLS (protocolo TLS). O TLS é um protocolo padrão do setor que garante conexões de rede criptografadas entre o servidor de banco de dados e os aplicativos cliente. O TLS é um protocolo atualizado de SSL (protocolo SSL).

O Banco de Dados do Azure para PostgreSQL - Servidor flexível só dá suporte a conexões criptografadas usando o protocolo TLS. Todas as conexões de entrada com o TLS 1.0 e o TLS 1.1 serão negadas.

Próximas etapas

- Saiba como criar um servidor flexível com acesso privado ([integração VNet](#)) no [portal do Azure](#) ou na [CLI do Azure](#).
- Saiba como criar um servidor flexível com acesso público ([endereços IP permitidos](#)) no [portal do Azure](#) ou na [CLI do Azure](#).

Limites no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

25/05/2021 • 6 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

As seções a seguir descrevem a capacidade e os limites funcionais no serviço de banco de dados. Para saber mais sobre as camadas de recurso (computação, memória e armazenamento), confira o artigo [computação e armazenamento](#).

Número máximo de conexões

O número máximo de conexões por tipo de preço e vCores é mostrado abaixo. O sistema do Azure exige três conexões para monitorar o Banco de Dados do Azure para PostgreSQL – Servidor Flexível.

NOME DO SKU	VCORES	TAMANHO DA MEMÓRIA	MÁXIMO DE CONEXÕES	MÁXIMO DE CONEXÕES DE USUÁRIO
Com capacidade de intermitência				
B1ms	1	2 GiB	50	47
B2s	2	4 GiB	100	97
Uso Geral				
D2s_v3	2	8 GiB	214	211
D4s_v3	4	16 GiB	429	426
D8s_v3	8	32 GiB	859	856
D16s_v3	16	64 GiB	1718	1715
D32s_v3	32	128 GiB	3437	3434
D48s_v3	48	192 GiB	5.000	4997
D64s_v3	64	256 GiB	5.000	4997
Otimizado para memória				
E2s_v3	2	16 GiB	1718	1715

NOME DO SKU	VCORES	TAMANHO DA MEMÓRIA	MÁXIMO DE CONEXÕES	MÁXIMO DE CONEXÕES DE USUÁRIO
E4s_v3	4	32 GiB	3437	3434
E8s_v3	8	64 GiB	5.000	4997
E16s_v3	16	128 GiB	5.000	4997
E32s_v3	32	256 GiB	5.000	4997
E48s_v3	48	384 GiB	5.000	4997
E64s_v3	64	432 GiB	5.000	4997

Quando as conexões excederem o limite, você poderá receber o seguinte erro:

FATAL: já existem muitos clientes

IMPORTANT

Para obter a melhor experiência, é recomendável usar um pooler de conexão como o PgBouncer para gerenciar as conexões com eficiência.

Uma conexão PostgreSQL, pode ocupar cerca de 10 MB de memória mesmo estando ociosa. Ainda, o estabelecimento de novas conexões leva tempo. A maioria dos aplicativos solicita muitas conexões de curta duração, o que agrava a essa situação. O resultado é um número menor de recursos disponíveis para sua carga de trabalho real, o que leva à redução do desempenho. O pool de conexão pode ser usado para diminuir as conexões ociosas e reutilizar as conexões existentes. Para saber mais, acesse nossa [postagem no blog](#).

Limitações funcionais

Operações de dimensionamento

- A colocação em escala do armazenamento requer a reinicialização do servidor.
- O armazenamento do servidor só pode ser dimensionado em incrementos de 2x, confira [computação e armazenamento](#) para obter os detalhes.
- Atualmente, não há suporte para diminuir o tamanho de armazenamento do servidor.

Upgrade da versão do servidor

- Não há suporte para a migração automatizada entre versões de mecanismo de banco de dados principal. Se você quiser atualizar para a próxima versão principal, faça um [despejo e restaure](#) para um servidor que foi criado com a nova versão do mecanismo.

Armazenamento

- Após ser configurado, o tamanho do armazenamento não pode ser reduzido. Você deve criar um servidor com o tamanho de armazenamento desejado, executar o [despejo manual e restaurar](#) e migrar os bancos de dados para o novo servidor.
- O recurso de aumento automático de armazenamento não está disponível no momento. Monitore o uso e aumente o tamanho armazenamento.
- Quando o uso do armazenamento atinge 95% ou se a capacidade disponível é inferior a 5 GiB, o servidor é alternado automaticamente para o [modo somente leitura](#) para evitar erros associados a situações de

disco cheio.

- É recomendável definir regras de alerta para `storage used` ou `storage percent` quando determinados limites forem excedidos para que você possa tomar medidas como aumentar o tamanho do armazenamento. Por exemplo, você pode definir um alerta para o caso de a porcentagem de armazenamento exceder 80% de uso.

Rede

- Não há suporte para a movimentação para dentro e para fora da VNET no momento.
- Não há suporte para a combinação do acesso público com a implantação em uma VNET no momento.
- Não há suporte para regras de firewall na VNET, em vez disso, podem ser usados os grupos de segurança de rede.
- Os servidores de banco de dados de acesso público podem se conectar à Internet pública, por exemplo, por meio de `postgres_fdw` e esse acesso não pode ser restrito. Os servidores baseados em VNET podem ter acesso de saída restrito usando os Grupos de Segurança de Rede.

HA (Alta disponibilidade)

- A HA com Redundância de Zona não tem suporte para servidores com Capacidade de Intermitência no momento.
- O endereço IP do servidor de banco de dados muda quando o servidor faz o failover para a HA em espera. Certifique-se de usar o registro DNS em vez do endereço IP do servidor.
- Se a replicação lógica estiver configurada com um servidor flexível configurado com HA, no caso de um failover para o servidor em espera, os slots de replicação lógica não serão copiados para o servidor em espera.
- Para obter mais detalhes sobre a HA com redundância de zona incluindo as limitações, confira a página de [conceitos – documentação de HA](#).

Zonas de disponibilidade

- Não há suporte para mover servidores manualmente para uma zona de disponibilidade diferente no momento.
- A zona de disponibilidade do servidor em espera com HA não pode ser configurada manualmente.

Mecanismo Postgres, extensões e PgBouncer

- Não há suporte para Postgres 10 e anteriores. É recomendável usar a opção [Servidor Único](#) se você precisa de versões mais antigas do Postgres.
- O suporte à extensão está limitado às extensões `contrib` do Postgres no momento.
- O pooler de conexão PgBouncer integrado não está disponível no momento para servidores de banco de dados em uma VNET ou para servidores com capacidade de intermitência.

Operação parar/iniciar

- O servidor não pode permanecer interrompido por mais de sete dias.

Manutenção agendada

- Alterar a janela de manutenção menos de cinco dias antes de uma atualização planejada não vai afetar essa atualização. As alterações só entram em vigor na próxima manutenção agendada.

Fazendo o backup de um servidor

- Os backups são gerenciados pelo sistema, não há nenhuma maneira de executá-los manualmente no momento. Em vez disso, é recomendável o uso de `pg_dump`.
- Os backups são sempre completos e baseados em instantâneo (backups não diferenciais), possivelmente levando a uma utilização de armazenamento de backup mais alta. Observe que os logs de transação (logs de gravação antecipada – WAL) são separados dos backups completos/diferenciais e são arquivados continuamente.

Restaurando um servidor

- Ao usar o recurso de Restauração Pontual, o novo servidor é criado com as mesmas configurações de computação e armazenamento nas quais o servidor está baseado.
- Os servidores de banco de dados baseados em VNET são restaurados na mesma VNET quando restaurados de um backup.
- O novo servidor criado durante uma restauração não possui as regras de firewall existentes no servidor original. As regras de firewall devem ser configuradas separadamente para esse novo servidor.
- Não há suporte para restaurar um servidor eliminado.
- Não há suporte para restauração entre regiões.

Outros recursos

- Ainda não há suporte para a autenticação do Azure AD. É recomendável usar a opção [Servidor Único](#) se você precisa de autenticação do Azure AD.
- Ainda não há suporte para réplicas de leitura. É recomendável usar a opção [Servidor Único](#) se você precisa de réplicas de leitura.
- Não há suporte para a transferência de recursos para outra assinatura.

Próximas etapas

- Entenda [o que está disponível para opções de computação e armazenamento](#)
- Saiba mais sobre [Versões de Banco de Dados PostgreSQL com suporte](#)
- Veja [Como fazer backup e restaurar um servidor no Banco de Dados do Azure para PostgreSQL usando o Portal do Azure](#)

Extensões PostgreSQL no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

09/08/2021 • 10 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O PostgreSQL fornece a capacidade de estender a funcionalidade de seu banco de dados usando as extensões. As extensões agrupam vários objetos SQL relacionados em um pacote que pode ser carregado ou removido do seu banco de dados com um comando. Depois de carregadas no banco de dados, as extensões funcionam como recursos internos.

Como usar as extensões PostgreSQL

As extensões PostgreSQL devem ser instaladas no banco de dados para que você possa usá-las. Para instalar uma extensão específica, execute o comando [CREATE EXTENSION](#). Esse comando carrega os objetos empacotados em seu banco de dados.

O Banco de Dados do Azure para PostgreSQL dá suporte a um subconjunto de extensões de chave, conforme listado abaixo. Essas informações também estão disponíveis por meio da execução de `SHOW azure.extensions;`. As extensões não listadas neste documento não têm suporte no banco de dados do Azure para PostgreSQL - Servidor Flexível. Você não pode criar ou carregar a sua própria extensão no Banco de Dados do Azure para PostgreSQL.

Extensões do Postgres 13

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL - Servidores Flexíveis que têm a versão 13 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	3.1.1	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	3.1.1	Exemplo de conjunto de dados de padronizador de endereço dos EUA
amcheck	1.2	Funções para verificar a integridade da relação
bloom	1.0	Método de acesso bloom – índice baseado em arquivo de assinatura
btree_gin	1,3	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1.5	suporte para indexação de tipos de dados comuns no GiST

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
citext	1.6	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.4	tipo de dados para cubos multidimensionais
dblink	1.2	conecta-se a outros bancos de dados PostgreSQL de dentro de um banco de dados
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
dict_xsyn	1.0	Modelo de dicionário de pesquisa de texto para processamento estendido de sinônimo
earthdistance	1.1	calcula distâncias de grandes círculos na superfície da Terra
fuzzystrmatch	1.1	determina as semelhanças e a distância entre cadeias de caracteres
hstore	1.7	tipo de dados para armazenar conjuntos de pares (chave, valor)
intagg	1.1	Agregador de inteiros e enumerador. (Obsoleto)
intarray	1.3	suporte a funções, operadores e índice para matrizes 1-D de inteiros
isn	1.2	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.2	tipo de dados para estruturas semelhantes a árvores hierárquicas
pageinspect	1.8	inspeciona o conteúdo das páginas do banco de dados em um nível baixo
pg_buffercache	1.3	examina o cache do buffer compartilhado
pg_cron	1.3	Agendador de trabalhos para PostgreSQL
pg_freespacemap	1.2	examina o mapa de espaço livre (FSM)
pg_partman	4.5.0	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1.2	pré-aquece dados de relações

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
pg_stat_statements	1.8	acompanha as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1.5	medição de similaridade de texto e da pesquisa de índice com base em trigrama
pg_visibility	1.2	examina o mapa de visibilidade (VM) e as informações de visibilidade no nível da página
pgaudit	1.5	fornecer a funcionalidade de auditoria
pgcrypto	1.3	funções criptográficas
pgrowlocks	1.2	mostra informações de bloqueio no nível de linha
pgstattuple	1.5	mostra estatísticas no nível da tupla
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
postgis	3.1.1	Geometria de PostGIS, geografia
postgis_raster	3.1.1	Funções e tipos de rasterização de PostGIS
postgis_sfcgal	3.1.1	Funções SFCGAL do PostGIS
postgis_tiger_geocoder	3.1.1	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	3.1.1	Tipos e funções espaciais de topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
sslinfo	1.2	informações sobre certificados SSL
tsm_system_rows	1.0	Método TABLESAMPLE, que aceita o número de linhas como um limite
tsm_system_time	1.0	Método TABLESAMPLE, que aceita o tempo em milissegundos como um limite
unaccent	1.1	dicionário de pesquisa de texto que remove acentos
uuid-ossp	1.1	gera identificadores universais únicos (UUIDs)

Extensões do Postgres 12

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL - Servidores Flexíveis que têm a versão 12 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	3.0.0	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	3.0.0	Exemplo de conjunto de dados de padronizador de endereço dos EUA
amcheck	1.2	Funções para verificar a integridade da relação
bloom	1.0	Método de acesso bloom – índice baseado em arquivo de assinatura
btree_gin	1.3	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1.5	suporte para indexação de tipos de dados comuns no GiST
citext	1.6	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.4	tipo de dados para cubos multidimensionais
dblink	1.2	conecta-se a outros bancos de dados PostgreSQL de dentro de um banco de dados
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
dict_xsyn	1.0	Modelo de dicionário de pesquisa de texto para processamento estendido de sinônimo
earthdistance	1.1	calcula distâncias de grandes círculos na superfície da Terra
fuzzystrmatch	1.1	determina as semelhanças e a distância entre cadeias de caracteres
hstore	1.6	tipo de dados para armazenar conjuntos de pares (chave, valor)
intagg	1.1	Agregador de inteiros e enumerador. (Obsoleto)

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
intarray	1.2	suporte a funções, operadores e índice para matrizes 1-D de inteiros
isn	1.2	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.1	tipo de dados para estruturas semelhantes a árvores hierárquicas
pageinspect	1.7	inspeciona o conteúdo das páginas do banco de dados em um nível baixo
pg_buffercache	1.3	examina o cache do buffer compartilhado
pg_cron	1.3	Agendador de trabalhos para PostgreSQL
pg_freespacemap	1.2	examina o mapa de espaço livre (FSM)
pg_partman	4.5.0	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1.2	pré-aquece dados de relações
pg_stat_statements	1.7	acompanha as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1.4	medição de similaridade de texto e da pesquisa de índice com base em trigramas
pg_visibility	1.2	examina o mapa de visibilidade (VM) e as informações de visibilidade no nível da página
pgaudit	1.4	fornecer a funcionalidade de auditoria
pgcrypto	1.3	funções criptográficas
pgrowlocks	1.2	mostra informações de bloqueio no nível de linha
pgstattuple	1.5	mostra estatísticas no nível da tupla
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
postgis	3.0.0	Geometria de PostGIS, geografia
postgis_raster	3.0.0	Funções e tipos de rasterização de PostGIS

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
postgis_sfsgal	3.0.0	Funções SFCGAL do PostGIS
postgis_tiger_geocoder	3.0.0	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	3.0.0	Tipos e funções espaciais de topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
sslinfo	1.2	informações sobre certificados SSL
tsm_system_rows	1.0	Método TABLESAMPLE, que aceita o número de linhas como um limite
tsm_system_time	1.0	Método TABLESAMPLE, que aceita o tempo em milissegundos como um limite
unaccent	1.1	dicionário de pesquisa de texto que remove acentos
uuid-ossp	1.1	gera identificadores universais únicos (UUIDs)

Extensões do Postgres 11

As extensões a seguir estão disponíveis nos servidores do Banco de Dados do Azure para PostgreSQL - Servidores Flexíveis que têm a versão 11 do Postgres.

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
address_standardizer	2.5.1	Usado para analisar um endereço em elementos constituintes.
address_standardizer_data_us	2.5.1	Exemplo de conjunto de dados de padronizador de endereço dos EUA
amcheck	1.1	Funções para verificar a integridade da relação
bloom	1.0	Método de acesso bloom – índice baseado em arquivo de assinatura
btree_gin	1.3	suporte para indexação de tipos de dados comuns no GIN
btree_gist	1.5	suporte para indexação de tipos de dados comuns no GiST

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
citext	1.5	tipo de dados para cadeias de caracteres que não diferenciam maiúsculas de minúsculas
cube	1.4	tipo de dados para cubos multidimensionais
dblink	1.2	conecta-se a outros bancos de dados PostgreSQL de dentro de um banco de dados
dict_int	1.0	modelo de dicionário de pesquisa de texto para inteiros
dict_xsyn	1.0	Modelo de dicionário de pesquisa de texto para processamento estendido de sinônimo
earthdistance	1.1	calcula distâncias de grandes círculos na superfície da Terra
fuzzystrmatch	1.1	determina as semelhanças e a distância entre cadeias de caracteres
hstore	1.5	tipo de dados para armazenar conjuntos de pares (chave, valor)
intagg	1.1	Agregador de inteiros e enumerador. (Obsoleto)
intarray	1.2	suporte a funções, operadores e índice para matrizes 1-D de inteiros
isn	1.2	tipos de dados para padrões de numeração de produtos internacionais
ltree	1.1	tipo de dados para estruturas semelhantes a árvores hierárquicas
pageinspect	1.7	inspeciona o conteúdo das páginas do banco de dados em um nível baixo
pg_buffercache	1.3	examina o cache do buffer compartilhado
pg_cron	1.3	Agendador de trabalhos para PostgreSQL
pg_freespacemap	1.2	examina o mapa de espaço livre (FSM)
pg_partman	4.5.0	Extensão usada para gerenciar tabelas particionadas por hora ou ID
pg_prewarm	1.2	pré-aquece dados de relações

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
pg_stat_statements	1.6	acompanha as estatísticas de execução de todas as instruções SQL executadas
pg_trgm	1.4	medição de similaridade de texto e da pesquisa de índice com base em trigrama
pg_visibility	1.2	examina o mapa de visibilidade (VM) e as informações de visibilidade no nível da página
pgaudit	1.3.1	fornecer a funcionalidade de auditoria
pgcrypto	1.3	funções criptográficas
pgrowlocks	1.2	mostra informações de bloqueio no nível de linha
pgstattuple	1.5	mostra estatísticas no nível da tupla
plpgsql	1.0	Linguagem de procedimento PL/pgSQL
postgis	2.5.1	Funções e tipos espaciais de geometria, geografia e rasterização do PostGIS
postgis_sfcgal	2.5.1	Funções SFCGAL do PostGIS
postgis_tiger_geocoder	2.5.1	Geocodificador PostGIS Tiger e geocodificador reverso
postgis_topology	2.5.1	Tipos e funções espaciais de topologia do PostGIS
postgres_fdw	1.0	wrapper de dados externos para servidores PostgreSQL remotos
sslinfo	1.2	informações sobre certificados SSL
tablefunc	1.0	funções que manipulam tabelas inteiras, incluindo a tabela de referência cruzada
tsm_system_rows	1.0	Método TABLESAMPLE, que aceita o número de linhas como um limite
tsm_system_time	1.0	Método TABLESAMPLE, que aceita o tempo em milissegundos como um limite
unaccent	1.1	dicionário de pesquisa de texto que remove acentos

EXTENSÃO	VERSÃO DA EXTENSÃO	DESCRIÇÃO
uuid-ossp	1,1	gera identificadores universais únicos (UUIDs)

dblink e postgres_fdw

Com [dblink](#) e [postgres_fdw](#), você pode se conectar de um servidor PostgreSQL a outro ou a outro banco de dados no mesmo servidor. O servidor flexível dá suporte a conexões de entrada e de saída para qualquer servidor PostgreSQL. O servidor de envio precisa permitir conexões de saída para o servidor de recebimento. Dessa forma, o servidor de recebimento precisa permitir conexões do servidor de envio por meio de seu firewall.

É recomendável implantar seus servidores com [Integração VNet](#) se você planeja usar essas duas extensões. Por padrão, a integração VNet permite conexões entre servidores na VNET. Você também pode optar por usar [grupos de segurança de rede de VNet](#) para personalizar o acesso.

pg_prewarm

A extensão pg_prewarm carrega dados relacionais no cache. O pré-aquecimento dos caches significa que as consultas têm tempos de resposta melhores na primeira execução após uma reinicialização. A funcionalidade de preaquecimento automático não está disponível no momento no Banco de Dados do Azure para PostgreSQL – Servidor Flexível.

pg_cron

[pg_cron](#) é um agendador de trabalhos simples e baseado em CRON para PostgreSQL, que é executado dentro do banco de dados como uma extensão. A extensão pg_cron pode ser usada para executar tarefas de manutenção agendadas em um banco de dados do PostgreSQL. Por exemplo, você pode executar o vácuo periódico de uma tabela ou remover trabalhos de dados antigos.

`pg_cron` pode executar vários trabalhos em paralelo, mas no máximo uma instância de um trabalho é executada por vez. Se uma segunda execução for iniciada antes da conclusão da primeira, a segunda execução ficará fila e será iniciada assim que a primeira for concluída. Isso garante que os trabalhos sejam executados exatamente conforme agendado, e não sejam executados simultaneamente com eles próprios.

Alguns exemplos:

Para excluir dados antigos no sábado às 3h30 (GMT)

```
SELECT cron.schedule('30 3 * * 6', $$DELETE FROM events WHERE event_time < now() - interval '1 week'$$);
```

Para executar o vácuo todos os dias às 10h (GMT)

```
SELECT cron.schedule('0 10 * * *', 'VACUUM');
```

Para cancelar o agendamento de todas as tarefas do pg_cron

```
SELECT cron.unschedule(jobid) FROM cron.job;
```

pg_stat_statements

A extensão `pg_stat_statements` é pré-carregada em cada servidor flexível do Banco de Dados do Azure para PostgreSQL para fornecer a você formas de acompanhar as estatísticas de execução das instruções SQL. A configuração `pg_stat_statements.track`, que controla quais instruções são contadas por extensão, tem `top` como padrão, que significa que todas as instruções emitidas diretamente por clientes são rastreadas. Dois outros níveis de rastreamento são `none` e `all`. Essa configuração é configurável como um parâmetro de servidor.

Há um equilíbrio entre as informações de execução de consulta fornecida por `pg_stat_statements` e o impacto no desempenho do servidor, que registra cada instrução SQL. Se você não está usando ativamente a extensão `pg_stat_statements`, recomendamos que você defina `pg_stat_statements.track` como `none`. Observe que alguns serviços de monitoramento de terceiros podem depender de `pg_stat_statements` para fornecer informações de desempenho de consulta, para confirmar se este é o caso para você ou não.

Próximas etapas

Se você não vir uma extensão que gostaria de usar, fale conosco. Vote em solicitações existentes ou crie comentários e solicitações em nosso [fórum de comentários](#).

Manutenção agendada no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

09/08/2021 • 3 minutes to read

O Banco de Dados do Azure para PostgreSQL – Servidor flexível realiza manutenções periódicas para manter seu banco de dados gerenciado seguro, estável e atualizado. Durante as manutenções, o servidor obtém novos recursos, atualizações e patches.

IMPORTANT

Banco de Dados do Azure para PostgreSQL – O servidor flexível está em versão prévia

Selecionando uma janela de manutenção

Você pode agendar uma manutenção durante um dia específico da semana e um período dentro desse dia. Ou pode permitir que o sistema escolha um dia e um período para você automaticamente. Seja como for, o sistema alertará você cinco dias antes de executar qualquer manutenção. O sistema também avisará quando a manutenção for iniciada e quando for concluída com êxito.

As notificações sobre as próximas manutenções agendadas podem ser:

- Enviadas por email para um endereço específico
- Enviadas por email para uma função do Azure Resource Manager
- Enviadas em um SMS (mensagem de texto) para dispositivos móveis
- Envio por push como notificação para um aplicativo do Azure
- Entregue como mensagem de voz

Ao especificar preferências para o agendamento de manutenção, você pode escolher um dia da semana e uma janela de tempo. Se você não especificar, o sistema escolherá os horários entre 23h e 7h no horário da região do servidor. Você pode definir agendamentos diferentes para cada servidor flexível em sua assinatura do Azure.

IMPORTANT

Normalmente, há pelo menos 30 dias entre os eventos de manutenção agendados bem-sucedidos para um servidor.

No entanto, no caso de uma atualização de emergência crítica, como uma vulnerabilidade grave, a janela de notificação pode ter menos de cinco dias. A atualização crítica pode ser aplicada ao seu servidor, mesmo se uma manutenção agendada bem-sucedida tiver sido realizada nos últimos 30 dias.

Você pode atualizar as configurações de agendamento a qualquer momento. Se houver uma manutenção agendada para seu Servidor flexível e você atualizar as preferências de agendamento, a distribuição atual continuará conforme agendado e a alteração das configurações de agendamento entrará em vigor após sua conclusão bem-sucedida na próxima manutenção agendada.

Você pode definir agendamento gerenciado pelo sistema ou agendamento personalizado para cada servidor flexível em sua assinatura do Azure.

- Com o agendamento personalizado, você pode especificar a janela de manutenção para o servidor escolhendo o dia da semana e uma janela de uma hora.

- Com o agendamento gerenciado pelo sistema, o sistema escolherá qualquer janela de uma hora entre 23h e 7h no horário da região do servidor.

Como parte das alterações da distribuição, aplicamos as atualizações aos servidores configurados com o agendamento gerenciado pelo sistema primeiro, seguidos por servidores com agendamento personalizado após um intervalo mínimo de sete dias em determinada região. Se você pretende receber atualizações antecipadas em frota de servidores de ambiente de desenvolvimento e teste, recomendamos configurar o agendamento gerenciado pelo sistema para servidores usados no ambiente de desenvolvimento e teste. Isso permitirá que você receba a atualização mais recente primeiro em seu ambiente de desenvolvimento/teste para teste e avaliação para validação. Se você encontrar qualquer comportamento ou alteração interruptiva, terá tempo para solucionar antes que a mesma atualização seja distribuída para servidores de produção com agendamento gerenciado personalizado. A atualização começa a ser distribuída em servidores flexíveis de agendamento personalizado após sete dias e é aplicada ao servidor na janela de manutenção definida. No momento, não há nenhuma opção para adiar a atualização depois que a notificação é enviada. O agendamento personalizado é recomendado apenas para ambientes de produção.

Em casos raros, o evento de manutenção pode ser cancelado pelo sistema ou pode não ser concluído com êxito. Se a atualização falhar, ela será revertida e a versão anterior dos binários será restaurada. Nesses cenários de atualização com falha, você ainda pode experimentar a reinicialização do servidor durante a janela de manutenção. Se a atualização for cancelada ou falhar, o sistema criará e enviará uma notificação para você sobre o evento de manutenção cancelado ou com falha. A próxima tentativa de realizar a manutenção será agendada de acordo com as configurações de agendamento atuais, e você receberá uma notificação com cinco dias de antecedência.

Próximas etapas

- Saiba como [alterar o agendamento de manutenção](#)
- Saiba como [receber notificações sobre manutenções futuras](#) usando a Integridade do Serviço do Azure
- Saiba como [configurar alertas sobre eventos futuros de manutenção agendada](#)

PgBouncer - Banco de Dados do Azure para PostgreSQL - servidor flexível

21/05/2021 • 4 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Banco de Dados do Azure para PostgreSQL – o servidor flexível oferece [PgBouncer](#) como uma solução de pooling de conexões integrada. Esse é um serviço opcional que pode ser habilitado por servidor por banco de dados e tem suporte com acesso público e privado. O PgBouncer é executado na mesma máquina virtual que o servidor de banco de dados Postgres. O Postgres usa um modelo baseado em processo para conexões, o que torna caro manter muitas conexões ociosas. Portanto, o próprio Postgres é executado em restrições de recursos depois que o servidor executa mais de alguns milhares de conexões. O principal benefício do PgBouncer é melhorar conexões ociosas e conexões de curta duração no servidor de banco de dados.

PgBouncer usa um modelo mais leve que utiliza E/S assíncrona e usa apenas conexões Postgres reais quando necessário, ou seja, quando dentro de uma transação aberta ou quando uma consulta está ativa. Esse modelo pode dar suporte a milhares de conexões mais facilmente com baixa sobrecarga e permite o dimensionamento para até 10.000 conexões com baixa sobrecarga.

Quando habilitado, PgBouncer é executado na porta 6432 no servidor de banco de dados. Você pode alterar a configuração de conexão de banco de dados do aplicativo para usar o mesmo nome de host, mas altere a porta para 6432 para começar a usar PgBouncer e se beneficiar do dimensionamento de conexão ocioso aprimorado.

NOTE

O PgBouncer só tem suporte em camadas de computação de Uso Geral e Otimizado para memória.

Habilitando e configurando o PgBouncer

Para habilitar o PgBouncer, você pode navegar até a folha "Parâmetros do Servidor" no portal do Azure e pesquisar por "PgBouncer" e alterar a configuração pgbouncer.enabled para "true" para PgBouncer a ser habilitado. Não é necessário reiniciar o computador. No entanto, para definir outros parâmetros PgBouncer, consulte a seção limitações.

Você pode definir as configurações do PgBouncer com estes parâmetros:

NOME DO PARÂMETRO	DESCRIÇÃO	PADRÃO
pgbouncer.default_pool_size	Definir esse valor de parâmetro como o número de conexões por par usuário/banco de dados	50
pgBouncer.max_client_conn	Definir esse valor de parâmetro como o número mais alto de conexões de cliente para PgBouncer que você deseja dar suporte	5.000

NOME DO PARÂMETRO	DESCRIÇÃO	PADRÃO
pgBouncer.pool_mode	Definir esse valor de parâmetro como TRANSACTION para pooling de transações (que é a configuração recomendada para a maioria das cargas de trabalho).	TRANSACTION
pgBouncer.min_pool_size	Adicione mais conexões de servidor ao pool se estiver abaixo desse número.	0 (desabilitado)
pgBouncer.stats_users	Opcional. Definir esse valor de parâmetro como o nome de um usuário existente, para poder fazer logon no banco de dados de estatísticas PgBouncer especial (chamado "PgBouncer")	

NOTE

A atualização do PgBouncer será gerenciada pelo Azure.

Alternando seu aplicativo para usar PgBouncer

Para começar a usar PgBouncer, siga estas etapas:

1. Conecte-se ao servidor de banco de dados, mas use a porta **6432** em vez da porta regular 5432 – verifique se essa conexão funciona

```
psql "host=myPgServer.postgres.database.azure.com port=6432 dbname=postgres user=myUser password=myPassword sslmode=require"
```

2. Teste seu aplicativo em um ambiente de qualidade em relação ao PgBouncer, para garantir que você não tenha problemas de compatibilidade. O projeto PgBouncer fornece uma matriz de compatibilidade e é recomendável usar **o pool de transações** para a maioria dos usuários:
<https://www.PgBouncer.org/features.html#sql-feature-map-for-pooling-modes>.

3. Altere seu aplicativo de produção para se conectar à porta **6432** em vez de 5432 e monitore quaisquer erros do lado do aplicativo que possam apontar para problemas de compatibilidade.

NOTE

Mesmo que você tenha habilitado o PgBouncer, ainda poderá se conectar ao servidor de banco de dados diretamente pela porta 5432 usando o mesmo nome de host.

PgBouncer em alta disponibilidade com redundância de zona

Em servidores configurados com alta disponibilidade com redundância de zona, o servidor primário executa o PgBouncer. Você pode se conectar ao PgBouncer do servidor primário pela porta 6432. Após um failover, o PgBouncer é reiniciado no modo de espera promovido recentemente, que é o novo servidor primário. Portanto, a cadeia de conexão do aplicativo permanece a mesma após o failover.

Usando PgBouncer com outros pools de conexão

Em alguns casos, você pode já ter um pool de conexões do lado do aplicativo ou ter o PgBouncer definido no lado do aplicativo, como um sidecar do AKS. Nesses casos, ainda pode ser útil utilizar o PgBouncer integrado, pois ele oferece benefícios de dimensionamento de conexão ociosa.

Utilizar um pool do lado do aplicativo junto com PgBouncer no servidor de banco de dados pode ser benéfico. Aqui, o pool do lado do aplicativo traz o benefício da latência de conexão inicial reduzida (pois a ida e volta inicial para inicializar a conexão é muito mais rápida) e o PgBouncer do lado do banco de dados fornece dimensionamento de conexão ociosa.

Limitações

- No momento, não há suporte para PgBouncer com a camada de computação do servidor com capacidade de intermitência.
- Se você alterar a camada de computação de Uso Geral ou Otimizado para memória para a camada Com capacidade de intermitência, perderá a funcionalidade PgBouncer.
- Sempre que o servidor é reiniciado durante operações de escala, failover de HA ou reinicialização, o PgBouncer também é reiniciado junto com a máquina virtual do servidor. Portanto, as conexões existentes devem ser reestabelecidas.
- Devido a um problema conhecido, o portal não mostra todos os parâmetros PgBouncer. Depois de habilitar PgBouncer e salvar o parâmetro, você precisa sair da tela Parâmetro (por exemplo, clique em Visão Geral) e voltar para a página Parâmetros.

Próximas etapas

- Saiba mais sobre [limitações de rede](#)
- Servidor Flexível [Visão geral](#)

Visão geral da continuidade dos negócios com o Banco de Dados do Azure para PostgreSQL – servidor flexível

09/08/2021 • 9 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Continuidade dos negócios no Banco de Dados SQL do Azure para PostgreSQL – Servidor Flexível refere-se a mecanismos, políticas e procedimentos que permitem que seu negócio continue operando em caso de interrupção, particularmente em sua infraestrutura de computação. Na maioria dos casos, o servidor flexível lidará com os eventos de interrupção que podem acontecer no ambiente de nuvem e manterá seus aplicativos e processos empresariais em execução. No entanto, há alguns eventos de interrupção que não podem ser tratados automaticamente, como:

- O usuário exclui ou atualiza acidentalmente uma linha em uma tabela.
- O terremoto causa uma interrupção de energia e desabilita um datacenter ou uma zona de disponibilidade de modo temporário.
- Aplicação de patch de banco de dados necessária para corrigir um problema de segurança ou bug.

O servidor flexível fornece recursos que protegem dados e reduzem o tempo de inatividade para seus bancos de dado de missão crítica no caso de eventos de tempo de inatividade planejado e não planejado. Criado com base na infraestrutura do Azure que já oferece resiliência e disponibilidade robustas, o servidor flexível tem recursos de continuidade de negócios que fornecem proteção adicional contra falha, cumprem os requisitos de tempo de recuperação e reduzem a exposição à perda de dados. Ao arquitetar seus aplicativos, você deve considerar a tolerância a tempo de inatividade, que é o RTO (objetivo de tempo de recuperação) e a exposição à perda de dados, que é o RPO (objetivo de ponto de recuperação). Por exemplo, seu banco de dados crítico para os negócios requer requisitos de tempo de atividade muito mais estritos em comparação a um banco de dados de teste.

IMPORTANT

O SLA (Contrato de Nível de Serviço) de tempo de atividade não é oferecido durante a versão prévia.

A tabela a seguir ilustra os recursos que o servidor flexível oferece.

RECURSO	DESCRÍÇÃO	CONSIDERAÇÕES
---------	-----------	---------------

RECURSO	DESCRÍÇÃO	CONSIDERAÇÕES
Backups automáticos	O servidor flexível executa automaticamente backups diários de seus arquivos de banco de dados e faz backup contínuo dos logs de transações. Os backups podem ser mantidos de sete dias até 35 dias. Você poderá restaurar o servidor de banco de dados para qualquer ponto no tempo dentro do período de retenção do backup. O RTO depende do tamanho dos dados a serem restaurados + o tempo para executar a recuperação de log. Pode ser de alguns minutos até 12 horas. Para obter mais detalhes, confira Conceitos – backup e restauração .	Os dados de backup permanecem dentro da região.
Alta disponibilidade com redundância de zona	O servidor flexível pode ser implantado com a configuração de HA (alta disponibilidade) com redundância de zona em que os servidores primário e em espera são implantados em duas zonas de disponibilidade diferentes dentro de uma região. Essa configuração de HA protege seus bancos de dados contra falhas no nível da zona e também ajuda a reduzir o tempo de inatividade do aplicativo durante eventos de tempo de inatividade planejado e não planejado. Os dados do servidor primário são replicados para a réplica em espera no modo síncrono. No caso de qualquer interrupção no servidor primário, o servidor passará por failover automaticamente na réplica em espera. O RTO na maioria dos casos deve ser menor que 120s. O RPO deve ser zero (sem perda de dados). Para saber mais, veja Conceitos – Alta disponibilidade .	Com suporte nas camadas de computação de uso geral e otimizado para memória. Disponível apenas em regiões em que há várias zonas disponíveis.
Discos gerenciados Premium	Os arquivos de banco de dados são armazenados em um armazenamento gerenciado Premium altamente durável e confiável. Isso fornece redundância de dados com três cópias de réplicas armazenadas em uma zona de disponibilidade com recursos de recuperação automática de dados. Para obter mais informações, confira a documentação de discos gerenciados .	Dados armazenados em uma zona de disponibilidade.

RECURSO	DESCRÍÇÃO	CONSIDERAÇÕES
Backup com redundância de zona	Os backups de servidor flexível são armazenados de modo automático e seguro em um armazenamento com redundância de zona em uma região. Durante uma falha no nível de zona em que o servidor é provisionado e, se o servidor não estiver configurado com redundância de zona, você ainda poderá restaurar seu banco de dados usando o ponto de restauração mais recente em uma zona diferente. Para obter mais informações, confira Conceitos – backup e restauração .	Aplicável somente em regiões em que várias zonas estão disponíveis.

Eventos de tempo de inatividade planejado

Abaixo estão alguns cenários de manutenção planejada. Normalmente, esses eventos geram até alguns minutos de inatividade e sem perda de dados.

CENÁRIO	PROCESSO
Escala de computação (iniciada pelo usuário)	Durante a operação de dimensionamento de computação, pontos de verificação ativos podem ser concluídos, as conexões de cliente são descarregadas, todas as transações não confirmadas são canceladas, o armazenamento é desanexado e, em seguida, é desligado. Um novo servidor flexível com o mesmo nome do servidor de banco de dados é provisionado com a configuração de computação dimensionada. O armazenamento é então anexado ao novo servidor e o banco de dados é iniciado, o que executa a recuperação, se necessário, antes de aceitar conexões de cliente.
Expansão do armazenamento (iniciado pelo usuário)	Quando a expansão de uma operação de armazenamento é iniciada, os pontos de verificação ativos têm permissão para serem concluídos, as conexões de cliente são descarregadas, todas as transações não confirmadas são canceladas e desligadas. O armazenamento é dimensionado para o tamanho desejado e então anexado ao novo servidor. Uma recuperação é executada, se necessário, antes de aceitar conexões de cliente. Observe que não há suporte para o dimensionamento vertical do tamanho do armazenamento.
Nova implantação de software (iniciada pelo Azure)	Novos recursos de distribuição ou correções de bugs ocorrem automaticamente como parte da manutenção planejada do serviço, e você pode agendar quando essas atividades ocorrerem. Para obter mais informações, verifique o portal .
Atualizações de versão secundárias (iniciadas pelo Azure)	O Banco de Dados do Azure para PostgreSQL corrige automaticamente os servidores de banco de dados para a versão secundária determinada pelo Azure. Isso acontece como parte da manutenção planejada do serviço. O servidor de banco de dados é reiniciado automaticamente com a nova versão secundária. Para obter mais informações, consulte a documentação . Você também pode verificar seu portal .

Quando o servidor flexível é configurado com **alta disponibilidade com redundância de zona**, o servidor flexível executa o dimensionamento e as operações de manutenção no servidor em espera primeiro. Para saber mais, veja [Conceitos – Alta disponibilidade](#).

Mitigação de tempo de inatividade não planejada

Os tempos de inatividade não planejados podem ocorrer como resultado de interrupções imprevisíveis, como falha de hardware subjacente, problemas de rede e bugs de software. Se o servidor de banco de dados configurado com alta disponibilidade falhar inesperadamente, a réplica em espera será ativada e os clientes poderão retomar as operações. Se não estiver configurado com alta disponibilidade (HA), se a tentativa de reinicialização falhar, um novo servidor de banco de dados será provisionado automaticamente. Embora um tempo de inatividade não planejado não possa ser evitado, o servidor flexível ajuda a reduzir o tempo de inatividade realizando automaticamente operações de recuperação sem exigir intervenção humana.

Tempo de inatividade não planejado: cenários de falha e recuperação de serviço

Abaixo estão alguns cenários de falha não planejados e o processo de recuperação.

CENÁRIO	PROCESSO DE RECUPERAÇÃO [NÃO HA]	PROCESSO DE RECUPERAÇÃO [HA]
Falha no servidor de banco de dados	<p>Se o servidor de banco de dados estiver inoperante, o Azure tentará reiniciar o servidor de banco de dados. Se isso falhar, o servidor de banco de dados será reiniciado em outro nó físico.</p> <p>O RTO (tempo de recuperação) depende de vários fatores, incluindo a atividade no momento da falha, como transação grande e o volume de recuperação a ser executada durante o processo de inicialização do servidor de banco de dados.</p> <p>Os aplicativos que usam os bancos de dados do PostgreSQL precisam ser criados para detectar e repetir as conexões e transações que falharam.</p>	<p>Se a falha do servidor de banco de dados for detectada, o servidor passará por failover para o servidor em espera, reduzindo o tempo de inatividade. Para obter mais informações, confira a página de conceitos de HA. O RTO deve estar entre 60 e 120s, com perda de dados zero.</p>
Falha de armazenamento	<p>Os aplicativos não detectam o impacto de problemas relacionados ao armazenamento, como uma falha de disco ou um dano a bloco físico. À medida que os dados são armazenados em três cópias, a cópia dos dados é atendida pelo armazenamento sobrevivente. O bloco de dados corrompido é automaticamente reparado e uma nova cópia dos dados é criada também de modo automático.</p>	<p>Para qualquer erro raro e não recuperável, como o armazenamento inteiro ficar inacessível, o servidor flexível faz failover para a réplica em espera para reduzir o tempo de inatividade. Para obter mais informações, confira a página de conceitos de HA.</p>

CENÁRIO	PROCESSO DE RECUPERAÇÃO [NÃO HA]	PROCESSO DE RECUPERAÇÃO [HA]
Erros de usuário/lógicos	<p>Para se recuperar de erros do usuário, como tabelas descartadas por engano ou dados atualizados de modo incorreto, você precisa executar uma PITR (recuperação pontual). Ao executar a operação de restauração, você especifica o ponto de restauração personalizado, que é a hora certa antes do erro.</p> <p>Se você quiser restaurar apenas um subconjunto de bancos de dados ou tabelas específicas, em vez de todos os bancos de dados no servidor de banco de dados, poderá restaurar o servidor de banco de dados em uma nova instância, exportar as tabelas por meio de <code>pg_dump</code> e usar <code>pg_restore</code> para restaurar essas tabelas em seu banco de dados.</p>	<p>Esses erros de usuário não são protegidos com alta disponibilidade, pois todas as alterações são replicadas para a réplica em espera de maneira síncrona. Você precisa executar a restauração pontual para se recuperar desses erros.</p>
Falha na zona de disponibilidade	<p>Para se recuperar de uma falha no nível de zona, você pode executar uma restauração pontual usando o backup e escolhendo um ponto de restauração personalizado com a hora mais recente para restaurar os dados mais recentes. Um novo servidor flexível será implantado em outra zona não afetada. O tempo necessário para a restauração depende do backup anterior e do volume de logs de transações a serem recuperados.</p>	<p>O servidor flexível faz failover automaticamente para o servidor em espera em 60 a 120s com perda de dados zero. Para obter mais informações, confira a página de conceitos de HA.</p>
Falha de região	<p>A réplica de leitura entre regiões e a restauração geográfica dos recursos de backup ainda não têm suporte na versão prévia.</p>	

IMPORTANT

Excluir servidores **não é possível** ser restaurado. Se você excluir o servidor, todos os bancos de dados que pertencem ao servidor também serão excluídos e não poderão ser recuperados. Use o [bloqueio de recursos do Azure](#) para ajudar a evitar a exclusão acidental do seu servidor.

Próximas etapas

- Saiba mais sobre [alta disponibilidade com redundância de zona](#)
- Saiba mais sobre [backup e recuperação](#)

Backup e restauração no Banco de Dados do Azure para PostgreSQL – Servidor flexível

09/08/2021 • 7 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Os backups formam uma parte essencial de qualquer estratégia de continuidade dos negócios. Eles ajudam a proteger os dados contra a corrupção ou exclusão acidental. O Banco de Dados do Azure para PostgreSQL – Servidor flexível faz o backup de seu servidor automaticamente e retém os backups por até 35 dias. Durante a restauração, você pode especificar a data e a hora em que deseja restaurar dentro do período de retenção. O tempo geral para restaurar e recuperar depende do tamanho dos arquivos de banco de dados e da quantidade de recuperação a ser executada.

Processo de backup no servidor flexível

O primeiro backup de instantâneo é agendado imediatamente após a criação do servidor flexível. Subsequentemente, um backup diário de instantâneo dos arquivos de dados é executado. Os backups são colocados em um armazenamento com redundância de zona em uma região. Os logs de transações (WAL – write ahead logs) também são arquivados continuamente, de modo que você poderá restaurar para a última transação confirmada. Esses backups de dados e logs permitem que você restaure um servidor pontualmente dentro de seu período de retenção de backup configurado. Todos os backups são criptografados usando a criptografia AES de 256 bits.

Se o banco de dados estiver configurado com alta disponibilidade, os instantâneos diários serão executados pelo primário e os backups de log contínuos serão executados pelo modo de espera.

IMPORTANT

Os backups não são executados em servidores interrompidos. No entanto, os backups são retomados quando o banco de dados é iniciado automaticamente após sete dias ou iniciado pelo usuário.

Os backups só podem ser usados para operações de restauração no Servidor flexível. Se você quiser exportar ou importar dados para o servidor flexível, use a metodologia de [despejo e restauração](#).

Retenção de backup

Os backups são mantidos com base na configuração do período de retenção de backup para o servidor. Você pode selecionar um período de retenção entre sete e 35 dias. O período de retenção padrão é de sete dias. Você pode definir o período de retenção durante a criação do servidor ou pode atualizá-lo posteriormente. Os backups são mantidos até mesmo para servidores interrompidos.

O período de retenção de backup determina até quando a restauração pontual pode ser feita, já que ele se baseia em backups disponíveis. O período de retenção de backup também pode ser tratado como uma janela de recuperação de uma perspectiva da restauração. Todos os backups necessários para executar uma restauração pontual dentro do período de retenção de backup são mantidos no armazenamento de backup. Por exemplo – se o período de retenção de backup for definido como sete dias, a janela de recuperação será considerada como os últimos sete dias. Nesse cenário, todos os dados e logs necessários para restaurar e recuperar o servidor nos últimos sete dias são mantidos.

Custo do armazenamento de backup

O servidor flexível fornece até 100% de seu armazenamento de servidor provisionado como armazenamento de backup sem custo adicional. Qualquer armazenamento de backup adicional usado será cobrado em GB por mês. Por exemplo, se você tiver provisionado um servidor com 250 GiB de armazenamento, terá 250 GiB de capacidade de armazenamento de backup sem custo adicional. Se o uso diário do backup for 25 GiB, você poderá ter até dez dias de armazenamento de backup gratuito. O consumo de armazenamento de backup que excede 250 GiB é cobrado de acordo com o [modelo de preços](#).

Você pode usar a métrica [Armazenamento de backup usado](#) no portal do Azure para monitorar o armazenamento de backup consumido por um servidor. A métrica Armazenamento de backup usada representa a soma do armazenamento consumido por todos os backups de banco de dados e backups de log retidos com base no período de retenção de backup definido para o servidor. Uma atividade transacional intensa no servidor pode fazer com que o uso do armazenamento de backup aumente, independentemente do tamanho total do banco de dados.

O principal meio de controlar o custo de armazenamento de backup é definindo o período de retenção de backup apropriado e escolhendo as opções de redundância de backup corretas para atender aos objetivos de recuperação desejados.

IMPORTANT

No momento, não há suporte para backups com redundância geográfica no servidor flexível.

Visão geral da restauração pontual

No Servidor flexível, executar uma restauração pontual cria um servidor por meio dos backups do servidor flexível na mesma região que o servidor de origem. Ele é criado com a configuração do servidor de origem para o tipo de preço, a geração da computação, o número de vCores, o tamanho do armazenamento, o período de retenção de backup e a opção de redundância de backup. Além disso, as marcas e configurações, como as configurações de VNET e firewall, são herdadas do servidor de origem.

IMPORTANT

Se você estiver restaurando um servidor flexível configurado com alta disponibilidade com redundância de zona, o servidor restaurado será configurado sem alta disponibilidade e na mesma região que o servidor primário.

Processo de restauração

Os arquivos de banco de dados físicos são restaurados primeiro dos backups de instantâneo para o local de dados do servidor. O backup apropriado que foi feito antes do ponto desejado é automaticamente escolhido e restaurado. Um processo de recuperação é iniciado usando arquivos WAL para colocar o banco de dados em um estado consistente.

Por exemplo, vamos supor que os backups são executados às 23h todas as noites. Se o ponto de restauração for de 15 de agosto de 2020 às 10h, o backup diário de 14 de agosto de 2020 será restaurado. O banco de dados será recuperado até 10h de 25 de agosto de 2020 usando o backup de log de transações entre as 23h de 24 de agosto e as 10h de 25 de agosto.

Confira [estas etapas](#) para restaurar o servidor de banco de dados.

IMPORTANT

As operações de restauração no servidor flexível criam um servidor de banco de dados e não substitui o servidor de banco de dados existente.

A Restauração pontual é útil em vários cenários. Por exemplo, quando um usuário exclui dados acidentalmente, descarta uma tabela ou um banco de dados importante, ou se um aplicativo acidentalmente substitui dados bons por dados inválidos devido a um defeito no aplicativo. Você poderá restaurar para a última transação devido ao backup contínuo de logs de transações.

Você pode escolher entre um ponto de restauração mais recente e um ponto de restauração personalizado.

- **Último ponto de restauração (agora)**: essa é a opção padrão que permite restaurar o servidor para o ponto no tempo mais recente.
- **Ponto de restauração personalizado**: esta opção permite que você escolha um ponto dentro do período de retenção definido para este servidor flexível. Por padrão, o horário mais recente em UTC será selecionado automaticamente e será útil se você quiser restaurar para a última transação confirmada para fins de teste. Você tem a opção de escolher outros dias e horários.

O tempo estimado de recuperação dependerá de vários fatores, incluindo tamanho do banco de dado, volume do log de transações a ser processado, largura de banda da rede e número total de bancos de dados em recuperação na mesma região e ao mesmo tempo. Normalmente, o tempo de recuperação geral demora de alguns minutos até algumas horas.

IMPORTANT

Excluir servidores **não é possível** ser restaurado. Se você excluir o servidor, todos os bancos de dados que pertencem ao servidor também serão excluídos e não poderão ser recuperados. Para proteger recursos do servidor, após a implantação, da exclusão acidental ou de alterações inesperadas, os administradores podem usar [bloqueios de gerenciamento](#).

Executar tarefas de pós-restauração

Após a restauração do banco de dados, você deverá executar as seguintes tarefas para colocar os usuários e os aplicativos novamente em execução:

- Se o novo servidor for usado para substituir o servidor original, redirecione clientes e aplicativos cliente para o novo servidor.
- Verifique se as regras de VNet e de firewall no nível do servidor adequadas estão em vigor para que os usuários se conectem. Essas regras não são copiadas do servidor original.
- A computação do servidor restaurado pode ser ampliada/reduzida conforme necessário.
- Verifique se permissões no nível do banco de dados e logins adequados estão em vigor.
- Configure os alertas conforme apropriado.
- Se você tiver restaurado o banco de dados configurado com alta disponibilidade e quiser configurar o servidor restaurado com alta disponibilidade, poderá seguir [as etapas](#).

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [alta disponibilidade com redundância de zona](#)
- Saiba [como restaurar](#)

Conceitos de alta disponibilidade no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

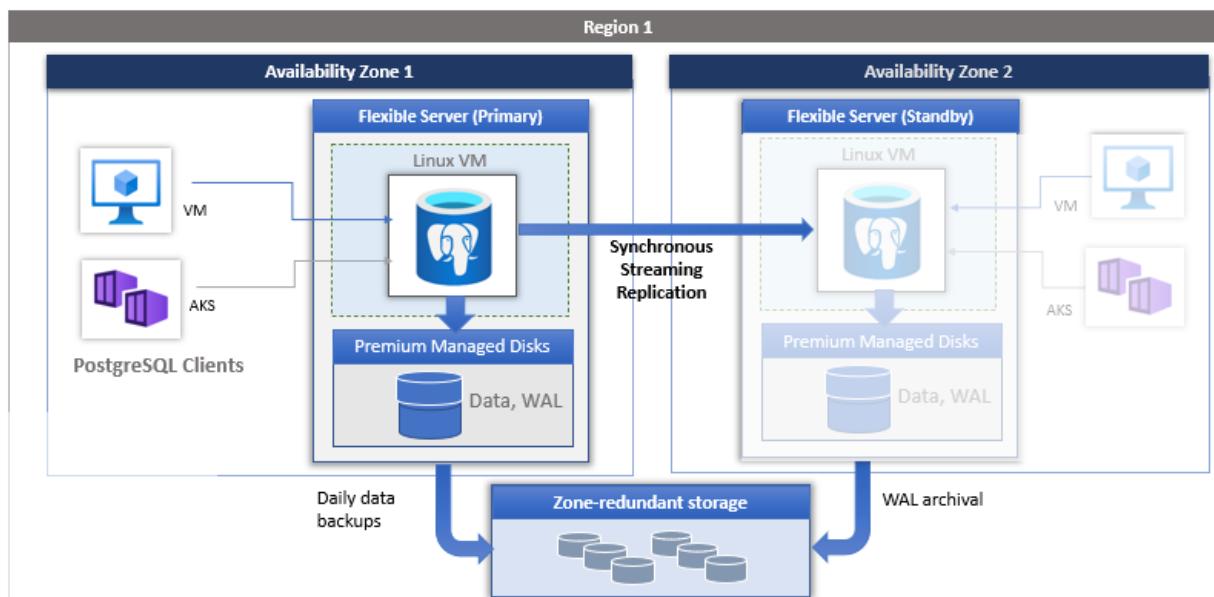
21/05/2021 • 7 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Banco de Dados do Azure para PostgreSQL – Servidor Flexível oferece configuração de alta disponibilidade com o recurso de failover automático usando a implantação do servidor com **redundância de zona**. Quando implantado em uma configuração com redundância de zona, o servidor flexível provisiona e gerencia automaticamente uma réplica em espera em uma zona de disponibilidade diferente. Usando a replicação de streaming do PostgreSQL, os dados são replicados para o servidor de réplica em espera no modo **síncrono**.

A configuração com redundância de zona habilita o recurso de failover automático sem perda de dados durante eventos planejados, como a operação de computação de escala iniciada pelo usuário, e também durante eventos não planejados, como falhas de hardware e de software subjacentes, falhas de rede e falhas de zona de disponibilidade.



Arquitetura de alta disponibilidade com redundância de zona

Você pode escolher a região e a zona de disponibilidade para implantar o servidor de banco de dados primário. Um servidor de réplica em espera é provisionado em uma zona de disponibilidade diferente com a mesma configuração do servidor primário, incluindo a camada de computação, o tamanho da computação, o tamanho do armazenamento e a configuração da rede. Os logs de transações são replicados no modo síncrono para a réplica em espera usando replicação de streaming do PostgreSQL. Os backups automáticos são executados periodicamente pelo servidor de banco de dados primário, enquanto os logs de transações são arquivados continuamente no armazenamento de backup da réplica em espera.

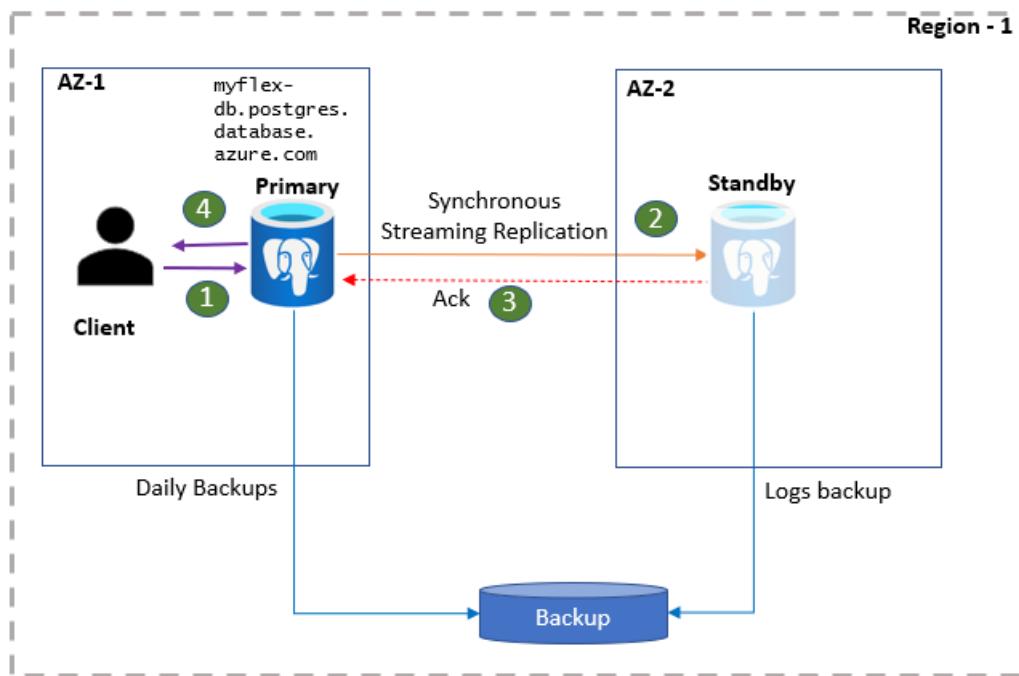
A integridade da configuração de alta disponibilidade é monitorada e relatada continuamente no portal. Os

status de alta disponibilidade com redundância de zona são listados abaixo:

STATUS	DESCRIÇÃO
Inicializando	No processo de criação de um servidor em espera
Replicando dados	Depois que o modo de espera é criado, ele é atualizado com o primário.
Íntegro	A replicação está em estado estável e íntegro.
Fazendo failover	O servidor de banco de dados está em processo de failover para o modo de espera.
Removendo do modo de espera	No processo de exclusão do servidor em espera.
Não habilitado	A alta disponibilidade com redundância de zona não está habilitada.

Operações de estado fixo

Os aplicativos cliente do PostgreSQL estão conectados ao servidor primário usando o nome do servidor do BD. As leituras de aplicativo são fornecidas diretamente do servidor primário, enquanto commits e gravações são confirmados no aplicativo somente depois que os dados são persistidos no servidor primário e na réplica em espera. Devido a esse requisito adicional de viagem de ida e volta, os aplicativos podem esperar uma latência elevada para gravações e commits. Você pode monitorar a integridade da alta disponibilidade no portal.



1. Os clientes se conectam ao servidor flexível e executam operações de gravação.
2. As alterações são replicadas para o site em espera.
3. O primário recebe a confirmação.
4. Gravações/commits são confirmados.

Processo de failover – tempos de inatividade planejados

Os eventos de tempo de inatividade planejado incluem atualizações de versão secundária e atualizações de

software periódicas agendadas do Azure. Quando configuradas em alta disponibilidade, essas operações são aplicadas primeiro à réplica em espera, enquanto os aplicativos continuam a acessar o servidor primário. Depois que a réplica em espera for atualizada, as conexões do servidor primário serão descarregadas e um failover será disparado, o que ativa a réplica em espera para ser a primária, com o mesmo nome do servidor de banco de dados. Os aplicativos clientes precisarão se reconectar com o mesmo nome do servidor de banco de dados ao novo servidor primário e poderão retomar suas operações. Um novo servidor em espera será estabelecido na mesma zona que o primário antigo.

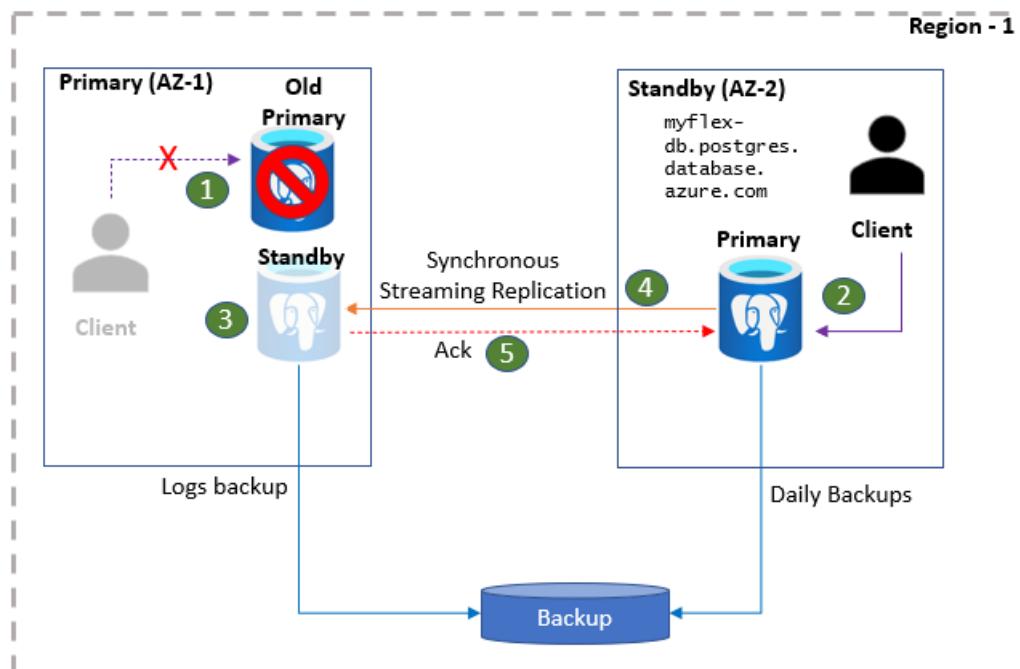
Para outras operações iniciadas pelo usuário, como computação de escala ou armazenamento de escala, as alterações são aplicadas primeiro no modo de espera, seguido pelo primário. Atualmente, as conexões não têm failover no modo de espera e, portanto, geram tempo de inatividade enquanto a operação é executada no servidor primário.

Reducir o tempo de inatividade planejado com a janela de manutenção gerenciada

Você pode agendar atividades de manutenção iniciadas pelo Azure escolhendo uma janela de 30 minutos em um dia de sua preferência em que as atividades nos bancos de dados devem ser baixas. As tarefas de manutenção do Azure, como aplicação de patch ou atualizações de versão secundária, ocorrerão durante essa janela. Para servidores flexíveis configurados com alta disponibilidade, essas atividades de manutenção são executadas na réplica em espera e são ativadas. Os aplicativos reconectam-se ao novo servidor primário e retomam suas operações enquanto um novo modo de espera é provisionado.

Processo de failover – tempos de inatividade não planejados

Interrupções não planejadas incluem bugs de software ou falhas do componente da infraestrutura que afetam a disponibilidade do banco de dados. Caso a indisponibilidade do servidor seja detectada pelo sistema de monitoramento, a replicação para a réplica em espera será severa, e a réplica em espera será ativada para ser o servidor de banco de dados primário. Os clientes podem se reconectar ao servidor de banco de dados usando a mesma cadeia de conexão e retomar suas operações. Espera-se que o tempo de failover geral seja de 60 a 120s. No entanto, dependendo da atividade no servidor de banco de dados primário no momento do failover, como transações grandes e tempo de recuperação, o failover poderá levar mais tempo.



1. O servidor de banco de dados primário está inoperante, e os clientes perdem a conectividade do banco de dados.
2. O servidor em espera é ativado para se tornar o novo servidor primário. O cliente se conecta ao novo servidor primário usando a mesma cadeia de conexão. Ter o aplicativo cliente na mesma zona que o servidor

- de banco de dados primário reduz a latência e melhora o desempenho.
3. O servidor em espera é estabelecido na mesma zona que o servidor primário antigo, e a replicação de streaming é iniciada.
 4. Depois que a replicação de estado fixo é estabelecida, gravações e commits do aplicativo cliente são confirmados depois que os dados são persistidos em ambos os locais.

Restauração em um momento determinado

Os servidores flexíveis configurados com alta disponibilidade replicam dados em tempo real para o servidor em espera a fim de mantê-los atualizados. Os erros do usuário no servidor primário, como um descarte acidental de uma tabela ou atualizações de dados incorretas, são reproduzidos de forma precisa na réplica em espera. Portanto, não é possível usar o modo de espera para se recuperar desses erros lógicos. Para se recuperar desses erros, você precisará executar a restauração pontual de backups. Usando o recurso de restauração pontual do servidor flexível, você poderá restaurar para o momento anterior ao erro. Para bancos de dados configurados com alta disponibilidade, um novo servidor de banco de dados será restaurado como um servidor flexível de zona única com um nome fornecido pelo usuário. Você pode exportar o objeto do servidor de banco de dados e importá-lo para seu servidor de banco de dados de produção. Da mesma forma, se você quiser clonar seu servidor de banco de dados para fins de teste e desenvolvimento ou restaurar para quaisquer outras finalidades, poderá executar restaurações pontuais.

Alta disponibilidade com redundância de zona – recursos

- A réplica em espera será implantada em uma configuração de VM exata igual ao servidor primário, incluindo vCores, armazenamento, configurações de rede (VNET, Firewall) etc.
- Capacidade de adicionar alta disponibilidade para um servidor de banco de dados existente.
- Capacidade de remover a réplica em espera desabilitando a alta disponibilidade.
- Capacidade de escolher sua zona de disponibilidade para o servidor de banco de dados primário.
- Capacidade de parar, iniciar e reiniciar os servidores de banco de dados primário e em espera.
- Os backups automáticos são executados do servidor de banco de dados primário e armazenados em um armazenamento com redundância de zona.
- Os clientes sempre se conectam ao servidor de banco de dados primário.
- Capacidade de reiniciar o servidor para selecionar qualquer alteração de parâmetro de servidor estático.
- Atividades de manutenção periódicas, como atualizações de versão secundária, acontecem primeiro em espera, e o serviço tem um failover para reduzir o tempo de inatividade.

Alta disponibilidade com redundância de zona – limitações

- Não há suporte para alta disponibilidade com a camada de computação com capacidade de intermitência.
- Há suporte para alta disponibilidade apenas em regiões em que várias zonas estão disponíveis.
- Devido à replicação síncrona para outra zona de disponibilidade, os aplicativos podem enfrentar latência elevada de gravação e commit.
- A réplica em espera não pode ser usada para consultas de leitura.
- Dependendo da carga de trabalho e da atividade no servidor primário, o processo de failover poderá levar mais de 120 segundos.

- Reiniciar o servidor de banco de dados primário também reiniciará a réplica em espera.
- Não há suporte para a configuração de réplicas de leitura adicionais.
- A configuração de tarefas de gerenciamento iniciadas pelo cliente não pode ser agendada durante a janela de manutenção gerenciada.
- Eventos planejados, como dimensionamento de computação e de armazenamento, ocorrem em espera primeiro e, em seguida, no servidor primário. O servidor não passou por failover para essas operações planejadas.
- Se a decodificação lógica ou a replicação lógica estiver configurada com um servidor flexível configurado com alta disponibilidade, no caso de um failover para o servidor em espera, os slots de replicação lógica não serão copiados para o servidor em espera.

Próximas etapas

- Saiba mais sobre a [continuidade dos negócios](#)
- Saiba como [gerenciar alta disponibilidade](#)
- Saiba mais sobre [backup e recuperação](#)

Monitorar métricas no Banco de Dados do Azure para PostgreSQL - servidor flexível

21/05/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Monitorar os dados dos seus servidores ajuda a solucionar problemas e otimizar sua carga de trabalho. O Banco de Dados do Azure para PostgreSQL oferece várias opções de monitoramento para fornecer insights sobre o comportamento do servidor.

Métricas

O Banco de Dados do Azure para PostgreSQL oferece várias métricas que fornecem informações sobre o comportamento dos recursos compatíveis com o servidor PostgreSQL. Cada métrica é emitida em uma frequência de um minuto e tem até [93 dias de histórico](#). É possível configurar alertas nas métricas. Outras opções incluem a configuração de ações automatizadas, execução de análises avançadas e arquivamento de histórico. Para obter mais informações, consulte a [Visão geral das métricas no Microsoft Azure](#).

Lista de métricas

As métricas a seguir estão disponíveis para o servidor flexível PostgreSQL:

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRÍÇÃO
active_connections	Conexões ativas	Contagem	O número de conexões ao seu servidor.
backup_storage_used	Backup do Microsoft Azure	Bytes	Quantidade de armazenamento de backup usado. A métrica representa a soma do armazenamento consumido por todos os backups de banco de dados, backups diferenciais e backups de log retidos com base no período de retenção de backup definido para o servidor. A frequência dos backups é gerenciada pelo serviço. Para o armazenamento com redundância geográfica, o uso de armazenamento de backup é o dobro do armazenamento com redundância local.
connections_failed	Conexões com falha	Contagem	Conexões com falha.

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRIÇÃO
connections_succeeded	Conexões com êxito	Contagem	Conexões com êxito.
cpu_credits_consumed	Créditos de CPU Consumidos	Contagem	Número de créditos usados pelo servidor flexível. Aplicável à camada com capacidade de intermitência.
cpu_credits_remaining	Créditos de CPU Restantes	Contagem	Número de créditos disponíveis para intermitência. Aplicável à camada com capacidade de intermitência.
cpu_percent	Porcentagem de CPU	Porcentagem	O percentual de CPU em uso.
disk_queue_depth	Profundidade da fila de disco	Contagem	Número de operações de E/S pendentes para o disco de dados.
iops	IOPS	Contagem	Número de operações E/S para disco por segundo.
maximum_used_transaction_ids	Máximo de IDs de transação usadas	Contagem	ID de transação máxima em uso.
memory_percent	Porcentagem de memória	Porcentagem	Porcentagem de memória em uso.
network_bytes_egress	Saída da rede	Bytes	Quantidade de tráfego de rede de saída.
network_bytes_ingress	Entrada na rede	Bytes	Quantidade de tráfego de rede de entrada.
read_iops	IOPS de leitura	Contagem	Número de operações de leitura de E/S de disco de dados por segundo.
read_throughput	Taxa de transferência de leitura	Bytes	Bytes lidos por segundo do disco.
storage_free	Livre para armazenamento	Bytes	A quantidade de espaço de armazenamento disponível.
storage_percent	Porcentagem de armazenamento	Percentual	Porcentagem de espaço de armazenamento usado. O armazenamento usado pelo serviço pode incluir os arquivos de banco de dados, os logs de transação e os logs do servidor.

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRIÇÃO
storage_used	Armazenamento usado	Bytes	Porcentagem de espaço de armazenamento usado. O armazenamento usado pelo serviço pode incluir os arquivos de banco de dados, os logs de transação e os logs do servidor.
txlogs_storage_used	Armazenamento de log de transações usado	Bytes	Quantidade de espaço de armazenamento usada pelos logs de transação.
write_throughput	Taxa de transferência de gravação	Bytes	Bytes gravados por segundo.
write_iops	IOPS de gravação	Contagem	Número de operações de gravação de E/S de disco de dados por segundo.

Logs do servidor

O Banco de Dados do Azure para PostgreSQL permite configurar e acessar os logs padrão do Postgres. Para saber mais sobre os logs, visite o [documento de conceitos de registro em log](#).

Faz logon no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

21/05/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Banco de Dados do Azure para PostgreSQL permite configurar e acessar os logs padrão do Postgres. Os logs podem ser usados para identificar, solucionar problemas e reparar erros de configuração e desempenho abaixo do ideal. As informações de registro em log que você pode configurar e acessar incluem erros, informações de consulta, registros de vácuo automático, conexões e pontos de verificação. (O acesso aos logs de transação não está disponível).

O log de auditoria é disponibilizado por meio de uma extensão Postgres, `pgaudit`. Para saber mais, visite o artigo [conceitos de auditoria](#).

Configurar o registro em log

Você pode configurar o log padrão do Postgres no seu servidor usando os parâmetros de registro em log. Para saber mais sobre os parâmetros de log do Postgres, visite as seções [Quando fazer um registro em log](#) e [O que registrar em log](#) na documentação do Postgres. A maioria, mas não todos, dos parâmetros de log do Postgres, estão disponíveis para configurar no Banco de Dados do Azure para PostgreSQL.

Para saber como configurar parâmetros no Banco de Dados do Azure para PostgreSQL, consulte a [documentação do portal](#) ou a [documentação da CLI](#).

NOTE

Configurar um alto volume de logs, por exemplo, registro de instruções em log, pode adicionar uma sobrecarga de desempenho significativa.

Acessando os logs

O Banco de Dados do Azure para PostgreSQL é integrado às configurações de diagnóstico do Azure Monitor. As configurações de diagnóstico permitem que você envie os logs do Postgres no formato JSON para os Logs do Azure Monitor para análise e alertas, Hubs de Eventos para streaming e Armazenamento do Microsoft Azure para arquivamento.

Formato de log

A tabela a seguir descreve os campos para o tipo **PostgreSQLLogs**. Dependendo do ponto de extremidade de saída escolhido, os campos incluídos e a ordem em que aparecem podem variar.

CAMPO	DESCRIÇÃO
TenantId	Sua ID de locatário
SourceSystem	Azure

CAMPO	DESCRIÇÃO
TimeGenerated [UTC]	Carimbo de data/hora quando o log foi gravado, em UTC
Tipo	Tipo do log. Sempre <code>AzureDiagnostics</code>
SubscriptionId	GUID para a assinatura a que o servidor pertence
ResourceGroup	Nome do grupo de recursos ao qual o servidor pertence
ResourceProvider	Nome do provedor de recursos. Sempre <code>MICROSOFT.DBFORPOSTGRESQL</code>
ResourceType	<code>Servers</code>
ResourceId	URI de recurso
Recurso	Nome do servidor
Category	<code>PostgreSQLLogs</code>
OperationName	<code>LogEvent</code>
errorLevel	Nível de log, exemplo: LOG, ERROR, NOTICE
Mensagem	Mensagem de log primária
Domínio	Versão do servidor, o exemplo: postgres-10
Detalhe	Mensagem de log secundária (se aplicável)
ColumnName	Nome da coluna (se aplicável)
SchemaName	Nome do esquema (se aplicável)
DatatypeName	Nome do tipo de dados (se aplicável)
LogicalServerName	Nome do servidor
_ResourceId	URI de recurso
Prefixo	Prefixo da linha de log

Próximas etapas

- Saiba mais sobre como [configurar e acessar logs](#).
- Saiba mais sobre o [preço do Azure Monitor](#).
- Saiba mais sobre [logs de auditoria](#)

Log de auditoria no Banco de Dados do Azure para PostgreSQL – Servidor flexível

21/05/2021 • 3 minutes to read

Os logs de auditoria de atividades do banco de dados no Banco de Dados do Azure para PostgreSQL – Servidor flexível estão disponíveis por meio da extensão de Auditoria do PostgreSQL: [pgAudit](#). O pgAudit fornece o log de auditoria detalhado de sessões e/ou objetos.

IMPORTANT

Banco de Dados do Azure para PostgreSQL – O servidor flexível está em versão prévia

Se você quiser logs no nível de recursos do Azure para operações como o dimensionamento de computação e armazenamento, veja o [Log de atividades do Azure](#).

Considerações sobre o uso

Por padrão, as instruções de log pgAudit são emitidas junto com suas instruções de log regulares usando o recurso de log padrão do Postgres. No Banco de Dados do Azure para PostgreSQL – Servidor flexível, você pode configurar todos os logs a serem enviados para o repositório de logs do Azure Monitor para análise posterior no Log Analytics. Se você habilitar Azure Monitor log de recursos, seus logs serão enviados automaticamente (no formato JSON) para o Armazenamento do Microsoft Azure, Hubs de Eventos e/ou logs de Azure Monitor, dependendo de sua escolha.

Para saber como configurar o log no Armazenamento do Microsoft Azure, nos Hubs de Eventos ou nos logs de Azure Monitor, visite a seção de logs de recursos do [artigo de logs do servidor](#).

Habilitar o pgAudit

Para habilitar o pgAudit, você precisa se conectar ao servidor usando um cliente (como psql) e habilitar a extensão pgAudit executando o seguinte comando:

```
CREATE EXTENSION pgaudit;
```

Configurações do pgAudit

O pgAudit permite que você configure o log de auditoria de sessão ou objeto. O [log de auditoria de sessão](#) emite logs detalhados de instruções executadas. O [log de auditoria de objeto](#) tem o escopo de auditoria para relações específicas. Você pode optar por configurar um ou ambos os tipos de registro em log.

NOTE

As configurações do pgAudit são especificadas globalmente e não podem ser especificadas em um nível de banco de dados ou de função.

Quando tiver [ativado o pgAudit](#), você poderá configurar os parâmetros dele para iniciar o registro em log. A [documentação do pgAudit](#) fornece a definição de cada parâmetro. Teste os parâmetros primeiro e confirme que

você está obtendo o comportamento esperado.

NOTE

Definir `pgaudit.log_client` como ATIVADO redirecionará os logs para um processo de cliente, como psql, em vez de serem gravados no arquivo. Essa configuração deve ser deixada desabilitada.

`pgaudit.log_level` é habilitado somente quando `pgaudit.log_client` está ativado.

NOTE

No Banco de Dados do Azure para PostgreSQL – Servidor flexível, `pgaudit.log` não pode ser definido usando um atalho de sinal (menos) `-`, conforme descrito na documentação do pgAudit. Todas as classes de instrução necessárias (LEITURA, GRAVAÇÃO etc.) devem ser especificadas individualmente.

Formato do log de auditoria

Cada entrada de auditoria é indicado `AUDIT:` próximo ao início da linha de log. O formato do restante da entrada é detalhado na [documentação do pgAudit](#).

Introdução

Para começar rapidamente, defina `pgaudit.log` como `WRITE` e abra seus logs de servidor para examinar a saída.

Exibir logs de auditoria

A maneira como você acessa os logs depende do ponto de extremidade escolhido. Para o Armazenamento do Microsoft Azure, veja o artigo sobre [conta de armazenamento de logs](#). Para os hubs de eventos, veja o artigo sobre [fluxos de logs do Azure](#).

Para logs de Azure Monitor, os logs são enviados para o espaço de trabalho selecionado. Os logs do Postgres usam o modo de coleta `AzureDiagnostics` para que possam ser consultados a partir da tabela `AzureDiagnostics`. Os campos na tabela são descritos abaixo. Saiba mais sobre como consultar e alertar na visão geral [Consulta de logs do Azure Monitor](#).

Você pode usar essa consulta para começar. Você pode configurar alertas com base em consultas.

Pesquisar em todas as entradas do pgAudit nos logs do Postgres por um servidor específico no último dia

```
AzureDiagnostics  
| where LogicalServerName_s == "myservername"  
| where TimeGenerated > ago(1d)  
| where Message contains "AUDIT:"
```

Próximas etapas

- [Saiba mais sobre o registro em log no Banco de Dados do Azure para PostgreSQL – Servidor flexível](#)
- [Saiba como configurar o registro em log no Banco de Dados do Azure para PostgreSQL – Servidor flexível e como acessar os logs](#)

Monitorar o desempenho com o Repositório de Consultas

21/05/2021 • 5 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – Servidor Flexível versão 11 e posteriores

O recurso de Repositório de Consultas no Banco de Dados do Azure para PostgreSQL fornece uma maneira de acompanhar o desempenho de consultas ao longo do tempo. O Repositório de Consultas simplifica a solução de problemas ajudando você a rapidamente localizar as consultas de execução mais longa e que consomem mais recursos. O Repositório de Consultas captura automaticamente um histórico das estatísticas de runtime e consultas e o retém para sua análise. Ele divide os dados por tempo para que você possa ver os padrões de uso temporais. Os dados de todos os usuários, bancos de dados e consultas são armazenados em um banco de dados chamado `azure_sys` na instância do Banco de Dados do Azure para PostgreSQL.

IMPORTANT

Não modifique o banco de dados `azure_sys` ou o esquema. Fazer isso impedirá que o Repositório de Consultas e os recursos de desempenho relacionados funcionem corretamente.

Habilitando o Repositório de Consultas

O Repositório de Consultas é um recurso que requer aceitação, portanto, ele não está habilitado em um servidor por padrão. O repositório de consultas é habilitado ou desabilitado globalmente para todos os bancos de dados em determinado servidor e não pode ser ativado ou desativado por banco de dados.

Habilitar o Repositório de Consultas usando o portal do Azure

- Entre no portal do Azure e selecione seu servidor do Banco de Dados do Azure para PostgreSQL.
- Selecione **Parâmetros de Servidor** na seção **Configurações** do menu.
- Pesquise o parâmetro `pg_qs.query_capture_mode`.
- Defina o valor como `TOP` ou `ALL` e clique em **Salvar**. Permita que o primeiro lote de dados persista no banco de dados `azure_sys` por até 20 minutos.

Informações no Repositório de Consultas

O Repositório de Consultas tem um repositório:

- Um repositório de estatísticas de runtime para manter as informações de estatísticas de execução de consulta.

Os cenários comuns para usar o Repositório de Consultas incluem:

- Determinação do número de vezes que uma consulta foi executada em uma determinada janela de tempo
- Comparar o tempo médio de execução de uma consulta entre janelas de tempo para ver grandes deltas
- Identificar consultas de execução mais longas nas últimas poucas horas. Para minimizar o uso de espaço, as estatísticas de execução de runtime no repositório de estatísticas de runtime são agregadas em uma janela de tempo fixa configurável. As informações contidas nesses repositórios podem ser consultadas usando exibições.

Acessar as informações do Repositório de Consultas

Os dados do Repositório de Consultas são armazenados no banco de dados `azure_sys` no servidor Postgres. A consulta a seguir retorna informações sobre consultas no Repositório de Consultas:

```
SELECT * FROM query_store.qs_view;
```

Opções de configuração

Quando o Repositório de Consultas está habilitado, ele salva dados em janelas de agregação de 15 minutos, até 500 consultas distintas por janela. As opções a seguir estão disponíveis para configurar os parâmetros do Repositório de Consultas.

PARÂMETRO	DESCRÍÇÃO	DEFAULT	RANGE
<code>pg_qs.query_capture_mode</code>	Define quais instruções são rastreadas.	nenhum	none, top, all
<code>pg_qs.max_query_text_length</code>	Define o comprimento máximo de consulta que pode ser salvo. Consultas mais longas serão truncadas.	6000	100 a 10 mil
<code>pg_qs.retention_period_in_days</code>	Define o período de retenção.	7	1 a 30
<code>pg_qs.track_utility</code>	Define se os comandos do utilitário são rastreados	on	on, off

Use o [portal do Azure](#) para obter ou definir um valor diferente para um parâmetro.

Exibições e funções

Exiba e gerencie o Repositório de Consultas usando as seguintes exibições e funções. Qualquer pessoa na função pública do PostgreSQL pode usar essas exibições para ver os dados no Repositório de Consultas. Essas exibições estão disponíveis somente no banco de dados `azure_sys`. Consultas são normalizadas examinando sua estrutura após a remoção de literais e constantes. Se duas consultas forem idênticas, exceto por valores literais, elas terão a mesma queryId.

`query_store.qs_view`

Essa exibição retorna todos os dados no Repositório de Consultas. Há uma linha para cada ID de banco de dados, ID de usuário e ID de consulta distinta.

NOME	TIPO	REFERÊNCIAS	DESCRÍÇÃO
<code>runtime_stats_entry_id</code>	BIGINT		ID da tabela <code>runtime_stats_entries</code>
<code>user_id</code>	oid	<code>pg_authid.oid</code>	OID do usuário que executou a instrução

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
db_id	oid	pg_database.oid	OID do banco de dados no qual a instrução foi executada
query_id	BIGINT		Código hash interno, computado da árvore de análise da instrução
query_sql_text	Varchar(10000)		Texto de uma instrução representativa. Consultas diferentes com a mesma estrutura são agrupadas. Este texto é o da primeira das consultas no cluster.
plan_id	BIGINT		ID do plano correspondente a essa consulta, ainda não disponível
start_time	timestamp		Consultas são agregadas por buckets de tempo: o período de um bucket é de 15 minutos por padrão. Essa é a hora de início correspondente ao bucket de tempo para esta entrada.
end_time	timestamp		Hora de término correspondente ao bucket de tempo para esta entrada.
chamadas	BIGINT		Número de vezes que a consulta foi executada
total_time	double precision		Tempo total de execução da consulta em milissegundos
min_time	double precision		Tempo mínimo de execução da consulta em milissegundos
max_time	double precision		Tempo máximo de execução da consulta em milissegundos
mean_time	double precision		Tempo médio de execução da consulta em milissegundos
stddev_time	double precision		Desvio padrão de tempo de execução da consulta em milissegundos

NOME	TIPO	REFERÊNCIAS	DESCRIÇÃO
rows	BIGINT		Número total de linhas recuperadas ou afetadas pela instrução
shared_blk_hit	BIGINT		Número total de ocorrências no cache do bloco compartilhado pela instrução
shared_blk_read	BIGINT		Número total de blocos compartilhados lidos pela instrução
shared_blk_dirtied	BIGINT		Número total de blocos compartilhados sujos pela instrução
shared_blk_written	BIGINT		Número total de blocos compartilhados gravados pela instrução
local_blk_hit	BIGINT		Número total de ocorrências no cache do bloco local pela instrução
local_blk_read	BIGINT		Número total de leituras de blocos locais pela instrução
local_blk_dirtied	BIGINT		Número total de blocos locais sujos pela instrução
local_blk_written	BIGINT		Número total de blocos locais gravados pela instrução
temp_blk_read	BIGINT		Número total de leituras de blocos temporários pela instrução
temp_blk_written	BIGINT		Número total de gravações de blocos temporários pela instrução
blk_read_time	double precision		Tempo total que a instrução passou lendo blocos em milissegundos (se track_io_timing estiver habilitado, caso contrário, zero)
blk_write_time	double precision		Tempo total que a instrução passou gravando blocos em milissegundos (se track_io_timing estiver habilitado, caso contrário, zero)

query_store.query_texts_view

Essa exibição retorna os dados de texto da consulta no Repositório de Consultas. Há uma linha para cada query_text distinto.

NOME	TIPO	DESCRIÇÃO
query_text_id	BIGINT	ID da tabela query_texts
query_sql_text	Varchar(10000)	Texto de uma instrução representativa. Consultas diferentes com a mesma estrutura são agrupadas. Este texto é o da primeira das consultas no cluster.

Funções

`qs_reset` descarta todas as estatísticas coletadas até o momento pelo Repositório de Consultas. Essa função só pode ser executada pela função de administrador de servidor.

`staging_data_reset` descarta todas as estatísticas coletadas na memória pelo Repositório de Consultas (isto é, os dados na memória que ainda não foram liberados para o banco de dados). Essa função só pode ser executada pela função de administrador de servidor.

Limitações e problemas conhecidos

- Se um servidor PostgreSQL tem o parâmetro `default_transaction_read_only` ativo, o Repositório de Consultas não irá capturar os dados.

Próximas etapas

- Saiba mais sobre [cenários em que o Repositório de Consultas pode ser especialmente útil](#).
- Saiba mais sobre as [melhores práticas para usar o Repositório de Consultas](#).

Cenários de uso do Repositório de Consultas - servidor flexível

21/05/2021 • 3 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – versões 11, 12 do servidor individual

Você pode usar o Query Store em uma ampla variedade de cenários, nos quais o acompanhamento e a manutenção do desempenho previsível da carga de trabalho são essenciais. Considere os seguintes exemplos:

- Identificar e ajustar consultas caras superior
- Testes de A/B
- Mantendo o desempenho estável durante as atualizações
- Identificando e melhorando as cargas de trabalho ad hoc

Identificar e ajustar consultas caras

Identificar as consultas de execução mais longas

Use o Repositório de Consultas exibições no banco de dados `azure_sys` do seu servidor para identificar rapidamente as consultas em execução mais longas. Essas consultas normalmente tendem a consumir muitos recursos. A otimização de suas consultas mais longas pode melhorar o desempenho, liberando recursos para uso por outras consultas em execução no seu sistema.

Consultas de destino com deltas de desempenho

O Query Store divide os dados de desempenho em janelas de tempo, para que você possa acompanhar o desempenho de uma consulta ao longo do tempo. Isso ajuda a identificar exatamente quais consultas estão contribuindo para um aumento no tempo total gasto. Como resultado, você pode fazer uma solução de problemas direcionada de sua carga de trabalho.

Ajuste de consultas caras

Quando você identifica uma consulta com desempenho abaixo do ideal, a ação executada depende da natureza do problema:

- Certifique-se de que as estatísticas estejam atualizadas para as tabelas subjacentes usadas pela consulta.
- Pense em reescrever as consultas caras. Por exemplo, aproveite a parametrização de consulta e reduza o uso de SQL dinâmico. Implemente a lógica ideal ao ler dados, como aplicar a filtragem de dados no lado do banco de dados, não no lado do aplicativo.

Testes de A/B

Use o Repositório de Consultas para comparar o desempenho da carga de trabalho antes e depois de uma alteração de aplicativo que você planeja introduzir ou antes e depois de uma migração. Exemplos de cenários de uso do Repositório de Consultas para avaliar o impacto do ambiente ou a alteração do aplicativo no desempenho da carga de trabalho:

- Migração entre versões do PostgreSQL.
- Implantando uma nova versão de um aplicativo.
- Adicionando recursos adicionais para o servidor.
- Criando índices ausentes em tabelas referenciadas por consultas caras.
- Migração de um servidor individual para o servidor Flex.

Em qualquer um desses cenários, aplique o seguinte fluxo de trabalho:

1. Execute sua carga de trabalho com o Query Store antes da mudança planejada para gerar uma linha de base de desempenho.
2. Aplique mudanças de aplicação no momento controlado no tempo.
3. Continue executando a carga de trabalho por tempo suficiente para gerar uma imagem de desempenho do sistema após a alteração.
4. Compare os resultados de antes e depois da alteração.
5. Decida se deseja manter a alteração ou reversão.

Identifique e melhore as cargas de trabalho ad hoc

Algumas cargas de trabalho não possuem consultas dominantes que você pode ajustar para melhorar o desempenho geral do aplicativo. Essas cargas de trabalho normalmente são caracterizadas com um número relativamente grande de consultas exclusivas, cada uma consumindo uma parte dos recursos do sistema. Cada consulta exclusiva é executada com pouca frequência, portanto, individualmente, seu consumo em runtime não é crítico. Por outro lado, como o aplicativo está gerando novas consultas o tempo todo, uma parte significativa dos recursos do sistema é gasta na compilação de consultas, o que não é o ideal. Normalmente, essa situação acontece se o aplicativo gerar consultas (em vez de usar procedimentos armazenados ou consultas parametrizadas) ou se depender de estruturas de mapeamento objeto-relacional que geram consultas por padrão.

Se você estiver no controle do código do aplicativo, considere reescrever a camada de acesso a dados para usar procedimentos armazenados ou consultas parametrizadas. No entanto, essa situação também pode ser melhorada sem alterações de aplicativo, forçando a parametrização de consulta para todo o banco de dados (todas as consultas) ou para os modelos de consulta individuais com o mesmo hash de consulta.

Próximas etapas

- Saiba mais sobre o [práticas recomendadas para usar a consulta Store](#)

Práticas recomendadas para o Repositório de Consultas -servidor flexível

21/05/2021 • 2 minutes to read

Aplica-se a: Banco de Dados do Azure para PostgreSQL – Servidor Flex versão 11, 12

Este artigo descreve as práticas recomendadas para o uso do Repositório de Consultas no Banco de Dados do Azure para PostgreSQL.

Definir o modo de captura de consulta ideal

Deixe que o Repositório de Consultas capture os dados que importam para você.

PG_QS.QUERY_CAPTURE_MODE	CENÁRIO
<i>Todos</i>	Analise sua carga de trabalho cuidadosamente em termos de todas as consultas e das respectivas frequências de execução e outras estatísticas. Identifique novas consultas na carga de trabalho. Detecte se consultas ad hoc são usadas para identificar oportunidades de parametrização automática ou pelo usuário. <i>All</i> acompanha um custo de consumo de recursos maior.
<i>Top</i>	Concentre sua atenção nas principais consultas – aquelas emitidas pelos clientes.
<i>Nenhum</i>	Se definido como nenhum, o Repositório de Consultas não capturará nenhuma consulta nova. Você já capturou um conjunto de consultas e a janela de tempo que você deseja investigar, e você deseja eliminar as distrações que outras consultas podem causar. <i>None</i> é adequado para teste e avaliação de desempenho de ambientes. <i>None</i> deve ser usado com cuidado, pois você pode perder a oportunidade de acompanhar e otimizar consultas novas importantes.

NOTE

`pg_qs.query_capture_mode` substitui `pgms_wait_sampling.query_capture_mode`. Se `pg_qs.query_capture_mode` for `none`, a configuração `pgms_wait_sampling.query_capture_mode` não terá efeito.

Manter os dados de que precisa

O parâmetro `pg_qs.retention_period_in_days` especifica, em dias, o período de retenção de dados para o Repositório de Consultas. Dados de consulta e estatísticas mais antigos são excluídos. Por padrão, o Repositório de Consultas é configurado para reter os dados por 7 dias. Evite manter dados históricos que você não planeja usar. Aumente o valor se você precisar manter dados por mais tempo.

Próximas etapas

- Saiba como obter ou definir parâmetros usando o [portal do Azure](#) ou a [CLI do Azure](#).

Replicação lógica e decodificação lógica no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

09/08/2021 • 4 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Banco de Dados do Azure para PostgreSQL - O Servidor Flexível dá suporte às seguintes metodologias lógicas de extração e replicação de dados:

1. **Replicação lógica**
 - a. Usando a [replicação lógica nativa](#) do PostgreSQL para replicar objetos de dados. A replicação lógica permite um controle refinado sobre a replicação de dados, incluindo a replicação de dados no nível da tabela.
2. **Decodificação lógica** implementada [decodificando](#) o conteúdo do WAL (log write-ahead).

Comparar a replicação lógica e a decodificação lógica

A replicação lógica e a decodificação lógica têm várias semelhanças. Ambas

- permitem que você replique dados do Postgres
- usar o [WAL \(log write-ahead\)](#) como origem das alterações
- usar [slots de replicação lógica](#) para enviar dados. Um slot representa um fluxo de alterações.
- usar a [propriedade REPLICA IDENTITY](#) de uma tabela para determinar quais alterações podem ser enviadas
- não replicar alterações de DDL

As duas tecnologias têm suas diferenças: a replicação lógica

- permite que você especifique uma tabela ou um conjunto de tabelas a serem replicadas
- replica dados entre instâncias do PostgreSQL

Decodificação lógica

- extrai alterações em todas as tabelas em um banco de dados
- não pode enviar dados diretamente entre instâncias do PostgreSQL.

Pré-requisitos para a replicação lógica e decodificação lógica

1. Acesse a página de parâmetros do servidor no portal.
2. Defina o parâmetro de servidor `wal_level` como `logical`.
3. Salve as alterações e reinicie o servidor para aplicar a alteração `wal_level`.
4. Confirme se a instância do PostgreSQL permite o tráfego de rede do seu recurso de conexão.
5. Conceda permissões de replicação de usuário administrador.

```
ALTER ROLE <adminname> WITH REPLICATION;
```

Usar a replicação lógica e a decodificação lógica

Replicação lógica nativa

A replicação lógica usa os termos 'publicador' e 'assinante'.

- O publicador é o banco de dados PostgreSQL **do** qual você está enviando dados.
- O assinante é o banco de dados PostgreSQL **para** o qual você está enviando dados.

Aqui está um código de exemplo que você pode usar para experimentar a replicação lógica.

1. Conectar ao banco de dados publicador. Criar uma tabela e adicionar alguns dados.

```
CREATE TABLE basic(id SERIAL, name varchar(40));
INSERT INTO basic(name) VALUES ('apple');
INSERT INTO basic(name) VALUES ('banana');
```

2. Crie uma publicação para a tabela.

```
CREATE PUBLICATION pub FOR TABLE basic;
```

3. Conecte-se ao banco de dados do assinante. Crie uma tabela com o mesmo esquema do publicador.

```
CREATE TABLE basic(id SERIAL, name varchar(40));
```

4. Crie uma assinatura que se conectará à publicação que você criou anteriormente.

```
CREATE SUBSCRIPTION sub CONNECTION 'host=<server>.postgres.database.azure.com user=<admin> dbname=<dbname> password=<password>' PUBLICATION pub;
```

5. Agora, você pode consultar a tabela no assinante. Você verá que ele recebeu dados do publicador.

```
SELECT * FROM basic;
```

Você pode adicionar mais linhas à tabela do publicador e exibir as alterações no assinante.

Se você não conseguir ver os dados, habilite o privilégio de logon para `azure_pg_admin` e verifique o conteúdo da tabela.

```
ALTER ROLE azure_pg_admin login;
```

Visite a documentação do PostgreSQL para entender mais sobre a [replicação lógica](#).

Decodificação lógica

A decodificação lógica pode ser consumida via protocolo de streaming ou interface do SQL.

Protocolo de streaming

O consumo de alterações usando o protocolo de streaming geralmente é preferível. Você pode criar seu próprio consumidor/conector ou usar uma ferramenta como [Debezium](#).

Visite a documentação do wal2json para [obter um exemplo usando o protocolo de streaming com pg_reclogical](#).

Interface do SQL

No exemplo a seguir, usamos a interface do SQL com o plug-in wal2json.

1. Crie um slot.

```
SELECT * FROM pg_create_logical_replication_slot('test_slot', 'wal2json');
```

2. Emita comandos SQL. Por exemplo:

```
CREATE TABLE a_table (
    id varchar(40) NOT NULL,
    item varchar(40),
    PRIMARY KEY (id)
);

INSERT INTO a_table (id, item) VALUES ('id1', 'item1');
DELETE FROM a_table WHERE id='id1';
```

3. Consuma as alterações.

```
SELECT data FROM pg_logical_slot_get_changes('test_slot', NULL, NULL, 'pretty-print', '1');
```

A saída terá a seguinte aparência:

```
{
  "change": [
  ]
}
{
  "change": [
    {
      "kind": "insert",
      "schema": "public",
      "table": "a_table",
      "columnnames": ["id", "item"],
      "columntypes": ["character varying(40)", "character varying(40)"],
      "columnvalues": ["id1", "item1"]
    }
  ]
}
{
  "change": [
    {
      "kind": "delete",
      "schema": "public",
      "table": "a_table",
      "oldkeys": {
        "keynames": ["id"],
        "keytypes": ["character varying(40)"],
        "keyvalues": ["id1"]
      }
    }
  ]
}
```

4. Descarte o slot quando terminar de usá-lo.

```
SELECT pg_drop_replication_slot('test_slot');
```

Visite a documentação do PostgreSQL para entender mais sobre a [replicação lógica](#).

Monitoramento

Você deve monitorar a decodificação lógica. Qualquer slot de replicação não utilizado deve ser descartado. Os slots mantêm os logs WAL do Postgres e os catálogos de sistema relevantes até que as alterações tenham sido lidas. Se o assinante ou consumidor falhar ou não tiver sido configurado corretamente, os logs não consumidos irão compilar e preencher o armazenamento. Além disso, os logs não consumidos aumentam o risco da delimitação de ID da transação. Ambas as situações podem fazer com que o servidor fique indisponível. Portanto, é essencial que os slots de replicação lógica sejam consumidos continuamente. Se um slot de replicação lógica não for mais usado, descarte-o imediatamente.

A coluna 'ativa' na exibição pg_replication_slots indicará se há um consumidor conectado a um slot.

```
SELECT * FROM pg_replication_slots;
```

[Defina alertas](#) sobre as **IDs de transação máximas usadas** e as métricas do servidor flexível de **Armazenamento utilizado** para notificá-lo quando os valores aumentarem os limites normais anteriores.

Limitações

- As limitações de [replicação lógica](#) se aplicam conforme documentado [aqui](#).
- **Rélicas de leitura** - As réplicas de leitura do Banco de Dados do Azure para PostgreSQL não têm suporte atualmente com servidores flexíveis.
- **Slots e failover de HA** - Os slots de replicação lógica no servidor primário não estão disponíveis no servidor em espera no AZ secundário. Isso se aplicará a você se o servidor usar a opção de alta disponibilidade com redundância de zona. No caso de um failover para o servidor em espera, os slots de replicação lógica não estarão disponíveis no modo de espera.

Próximas etapas

- Saiba mais sobre as [opções de rede](#)
- Saiba mais sobre as [extensões](#) disponíveis no servidor flexível
- Saiba mais sobre [alta disponibilidade](#)

Gerenciar um Banco de Dados do Azure para PostgreSQL - Servidor Flexível com o portal do Azure

09/08/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo mostra como gerenciar seu Banco de Dados do Azure para PostgreSQL - Servidor Flexível. As tarefas de gerenciamento incluem dimensionamento de computação e armazenamento, redefinição de senha de administrador e exibição de detalhes do servidor.

Entrar

Entre no [portal do Azure](#). Navegue até o recurso do servidor flexível no portal do Azure.

Dimensionar a computação e o armazenamento

Depois de criar um servidor, é possível realizar o dimensionamento para as várias [camadas de preço](#) à medida que suas necessidades mudam. Você também pode escalar ou reduzir verticalmente sua computação e sua memória aumentando ou diminuindo vCores.

NOTE

O armazenamento não pode ser dimensionado para um valor mais baixo.

1. No portal do Azure, selecione o servidor. Selecione **Computação + Armazenamento** na seção **Configurações**.
2. É possível alterar a **Camada de computação**, o **vCore** e o **Armazenamento** para dimensionar o servidor na mesma camada ou em uma camada de computação mais alta, aumentando o armazenamento ou os vCores para o valor desejado.

flex-pg-server | Compute + storage
Azure Database for PostgreSQL flexible server

Search (Ctrl+ /) <>

Compute + storage

Compute resources are pre-allocated and billed per minute based on vCores configured.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (4-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (4-64 vCores) - Best for workloads that require a high memory to CPU ratio

vCore: 1

Memory 2 GiB

Storage size (in GiB): 32

Storage cannot be scaled down

UP TO 120 AVAILABLE IOPS

Backups

Configure automatic server backups that can be used to restore your server to a point-in-time.

Retention period: 7

Monitoring

- Alerts
- Metrics
- Diagnostic settings
- Logs

IMPORTANT

- Não é possível diminuir o dimensionamento do armazenamento.
- O dimensionamento de vCores causa uma reinicialização do servidor.

3. Selecione **OK** para salvar as alterações.

Redefinir a senha de administrador

É possível alterar a senha da função de administrador no portal do Azure.

1. No portal do Azure, selecione o servidor. Na janela **Visão geral**, selecione **Redefinir senha**.
2. Insira uma nova senha e confirme-a. A caixa de texto informará os requisitos de complexidade da senha.

flex-pg-server flex-pg-server

Azure Database for PostgreSQL flexible server | PREVIEW

Search (Ctrl+ /) <>

Reset the password

Save Discard

Resource utilization (sumuth-flex-pg-server)

Time	Utilization (%)
100	100
90	90
80	80

Essentials

Resource group: **flexible-server**

Status: Available

Location: East US

Subscription: **team**

Subscription ID: [REDACTED]

Tags (change): Click here to add tags

Show data for last: 1 hour

3. Selecione **Salvar** para salvar a nova senha.

Excluir um servidor

Será possível excluir o servidor se ele não for necessário.

1. No portal do Azure, selecione o servidor. Na janela **Visão geral**, selecione **Excluir**.
2. Digite o nome do servidor na caixa de entrada para confirmar que o servidor correto será excluído.

The screenshot shows the Azure portal interface for managing a PostgreSQL flexible server named 'flex-pg-server'. On the left, the 'Overview' tab is selected, displaying details like Resource group (-flexible-server), Status (Available), Location (East US), and Subscription (team). On the right, a modal window titled 'Are you sure you want to delete...' is open, asking for confirmation to delete the server. It includes a warning message: 'Warning! Deleting sumuth-flex-pg-server is irreversible. The action you're about to take can't be undone. Going further will delete it and all the items in it permanently.' Below the warning, there is a text input field labeled 'TYPE THE AZURE DATABASE FOR POSTGRESQL FLEXIBLE SERVER NAME' with the value 'flex-pg-server' entered. At the bottom of the modal, there are buttons for 'Affected items' and 'Resource name'.

IMPORTANT

A exclusão de um servidor é irreversível.

This screenshot is identical to the one above, showing the Azure portal interface for managing a PostgreSQL flexible server named 'flex-pg-server'. The 'Overview' tab is selected, and a deletion confirmation modal is open. The warning message and the text input field both show 'flex-pg-server'. The 'Affected items' and 'Resource name' sections at the bottom of the modal are also present.

3. Selecione **Excluir**.

Próximas etapas

- Conceitos básicos de backup e restauração
- Ajustar e monitorar o servidor

Gerenciar um servidor flexível do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure

21/05/2021 • 4 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia.

Este artigo mostra como gerenciar seu servidor flexível implantado no Azure. As tarefas de gerenciamento incluem o dimensionamento da computação e do armazenamento, a redefinição de senhas do administrador e a visualização de detalhes do servidor.

Pré-requisitos

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Você precisará executar a CLI do Azure versão 2.0 ou posterior, localmente. Para ver a versão instalada, execute o comando `az --version`. Se você precisar instalar ou atualizar, confira [Instalar a CLI do Azure](#).

Entre na sua conta usando o comando `az login`.

```
az login
```

Selecione uma assinatura usando o comando `az account set`. Anote o valor de `id` da saída de `az login` para usá-lo como o valor para o argumento `subscription` no comando a seguir. Se tiver várias assinaturas, escolha aquela para cobrança do recurso. Para identificar todas suas assinaturas, use o comando `az account list`.

```
az account set --subscription <subscription id>
```

IMPORTANT

Se você ainda não criou um servidor flexível, precisará fazer isso para seguir este guia de instruções.

Dimensionar a computação e o armazenamento

Você pode dimensionar sua camada de computação, vCores e armazenamento facilmente usando o comando a seguir. Para obter uma lista de todas as operações de servidor que você pode executar, confira a visão geral de [az postgres flexible-server](#).

```
az postgres flexible-server update --resource-group myresourcegroup --name mydemoserver --sku-name Standard_D4ds_v3 --storage-size 6144
```

A seguir estão os detalhes dos argumentos no código anterior:

CONFIGURAÇÃO	VALOR DE EXEMPLO	DESCRIÇÃO
name	mydemoserver	Insira um nome exclusivo para o seu servidor. O nome do servidor pode conter apenas letras minúsculas, números e o caractere de hífen (-). Ele precisa conter de 3 a 63 caracteres.
resource-group	myresourcegroup	Forneça o nome do grupo de recursos do Azure.
sku-name	Standard_D4ds_v3	Insira o nome do tamanho e da camada de computação. O valor segue a convenção <i>Standard_{VM size}</i> abreviadamente. Confira os tipos de preço para obter mais informações.
storage-size	6144	Insira a capacidade de armazenamento do servidor em megabytes. O mínimo é 5120, aumentando em incrementos de 1024.

IMPORTANT

Não é possível reduzir verticalmente o armazenamento.

Gerenciar bancos de dados PostgreSQL em um servidor

Há vários aplicativos que você pode usar para conectar o servidor Banco de Dados do Azure para PostgreSQL. Se o computador cliente tiver o PostgreSQL instalado, você poderá usar uma instância local do [psql](#). Usaremos agora a ferramenta de linha de comando psql para nos conectarmos ao servidor do Banco de Dados do Azure para PostgreSQL.

- Execute o seguinte comando psql:

```
psql --host=<servername> --port=<port> --username=<user> --dbname=<dbname>
```

Por exemplo, o comando a seguir se conecta ao banco de dados padrão chamado **postgres** no seu servidor PostgreSQL **mydemoserver.postgres.database.azure.com** por meio das suas credenciais de acesso. Quando for solicitado, insira a **<server_admin_password>** que você escolheu.

```
psql --host=mydemoserver.postgres.database.azure.com --port=5432 --username=myadmin --dbname=postgres
```

Depois que você se conectar, a ferramenta psql exibirá um prompt do **postgres** no qual você digitará os comandos do SQL. Um aviso será exibido na saída de conexão inicial se a versão do psql que você está usando for diferente da versão no servidor do Banco de Dados do Azure para PostgreSQL.

Exemplo de saída psql:

```
psql (11.3, server 12.1)
WARNING: psql major version 11, server major version 12.
          Some psql features might not work.
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=>
```

TIP

Se o firewall não está configurado para permitir o endereço IP do seu cliente, ocorre o seguinte erro:

```
"psql: FATAL: nenhuma entrada pg_hba.conf para o host <IP address>, usuário "myadmin", banco de dados
"postgres", SSL em FATAL: Uma conexão SSL é necessária. Especifique as opções de SSL e tente novamente".
```

Confirme se o endereço IP do seu cliente é permitido nas regras de firewall.

2. Crie um banco de dados chamado **postgresdb** em branco digitando o seguinte comando no prompt:

```
CREATE DATABASE postgresdb;
```

3. No prompt, execute o seguinte comando para mudar as conexões para o banco de dados **postgresdb** recém-criado:

```
\c postgresdb
```

4. Digite **\q** e selecione Enter para sair do psql.

Nesta seção, você está conectado ao Banco de Dados do Azure para servidor PostgreSQL via psql e criou um banco de dados do usuário em branco.

Redefinir a senha do administrador

Você pode alterar a senha da função de administrador com este comando:

```
az postgres flexible-server update --resource-group myresourcegroup --name mydemoserver --admin-password
<new-password>
```

IMPORTANT

Escolha uma senha que tenha um mínimo de 8 caracteres e um máximo de 128 caracteres. A senha precisa conter caracteres de três das seguintes categorias:

- Letras maiúsculas do alfabeto inglês
- Letras minúsculas do alfabeto inglês
- Números
- Caractere não alfanumérico

Excluir um servidor

Para excluir o servidor flexível do Banco de Dados do Azure para PostgreSQL, execute o comando [az postgres flexible-server delete](#).

```
az postgres flexible-server delete --resource-group myresourcegroup --name mydemoserver
```

Próximas etapas

- [Entender os conceitos de backup e restauração](#)
- [Ajustar e monitorar o servidor](#)

Visão geral de conexão e consulta do banco de dados do Azure para PostgreSQL – Servidor Flexível

21/05/2021 • 2 minutes to read

O documento a seguir inclui links para exemplos que mostram como se conectar e consultar o Servidor Único do Banco de Dados do Azure para PostgreSQL. Este guia também inclui as recomendações e extensões de TLS que podem ser usadas para se conectar ao servidor nos idiomas com suporte abaixo.

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em **versão prévia**.

Inícios rápidos

GUIA DE INÍCIO RÁPIDO	DESCRIÇÃO
Pgadmin	Você pode usar o pgadmin para se conectar ao servidor, o que simplifica a criação, a manutenção e o uso de objetos de banco de dados.
psql no Azure Cloud Shell	Este artigo mostra como executar <code>psql</code> no Azure Cloud Shell para se conectar ao servidor e executar instruções para consultar, inserir, atualizar e excluir dados no banco de dados. Você pode executar o <code>psql</code> se ele estiver instalado em seu ambiente de desenvolvimento
Python	Este guia de início rápido demonstra como usar o Python para se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.
Django com Serviço de Aplicativo	Este tutorial demonstra como usar o Ruby para criar um programa para se conectar a um banco de dados e trabalhar com objetos de banco de dados para consultar dados.

Considerações sobre o TLS para a conectividade de banco de dados

O TLS é usado por todos os drivers que a Microsoft fornece ou dá suporte para conexão a bancos de dados no Banco de Dados SQL do Azure para PostgreSQL. Nenhuma configuração especial é necessária, mas estabeleça o TLS 1.2 para servidores recém-criados. Se você estiver usando o TLS 1.0 ou 1.1, é recomendável atualizar a versão do TLS para os servidores. Confira [Como configurar o TLS](#)

Extensões do PostgreSQL

O PostgreSQL fornece a capacidade de estender a funcionalidade de seu banco de dados usando as extensões. As extensões agrupam vários objetos SQL em um pacote que pode ser carregado ou removido do seu banco de dados com um comando. Depois de carregadas no banco de dados, as extensões funcionam como recursos internos.

- Extensões do Postgres 12
- Extensões do Postgres 11
- dblink e postgres_fdw
- pg_prewarm
- pg_stat_statements

Para obter mais detalhes, confira [Como usar as extensões do PostgreSQL em um servidor flexível](#).

Próximas etapas

- Migrar dados usando o despejo e a restauração
- Migrar dados usando a importação e a exportação

Gerenciar configurações de manutenção agendadas para o Banco de Dados do Azure para PostgreSQL – servidor flexível

21/05/2021 • 2 minutes to read

Você pode especificar opções de manutenção para cada servidor flexível em sua assinatura do Azure. As opções incluem o agendamento de manutenção e as configurações de notificação para eventos de manutenção futuros e concluídos.

IMPORTANT

Banco de Dados do Azure para PostgreSQL – o servidor flexível está em versão preliminar.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Um [Banco de Dados do Azure para PostgreSQL – servidor flexível](#)

Especificar opções de agendamento de manutenção

1. Na página do servidor PostgreSQL, no título **Configurações**, escolha **Manutenção** para abrir as opções de manutenção agendada.
2. A agenda padrão (gerenciada pelo sistema) é um dia da semana aleatório e uma janela de 60 minutos para o início da manutenção entre 23h e 7h, horário do servidor local. Se você quiser personalizar essa agenda, escolha **Personalizar agenda**. Em seguida, é possível selecionar um dia da semana de preferência e uma janela de 60 minutos para a hora de início da manutenção.

Notificações sobre eventos de manutenção agendada

É possível usar a Integridade do Serviço do Azure para [exibir notificações](#) sobre manutenções agendadas futuras e realizadas em seu servidor flexível. Também é possível [configurar](#) alertas na Integridade do Serviço do Azure para obter notificações sobre eventos de manutenção.

Próximas etapas

- Saiba mais sobre [Manutenção agendada no Banco de Dados do Azure para PostgreSQL – servidor flexível](#)
- Saiba mais sobre a [Integridade do Serviço do Azure](#)

Criar e gerenciar redes virtuais do Banco de Dados do Azure para PostgreSQL - Servidor flexível usando o portal do Azure

09/08/2021 • 3 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL dá suporte a dois tipos de métodos de conectividade de rede mutuamente exclusivos, para se conectar ao seu servidor flexível: As duas opções são:

- Acesso público (endereços IP permitidos)
- Acesso privado (Integração VNet)

Neste artigo, vamos enfocar a criação do servidor PostgreSQL com **Acesso privado (integração VNet)** usando o portal do Azure. Com Acesso privado (integração de VNet), é possível implantar o servidor flexível na [Rede Virtual do Azure](#). As Redes Virtuais do Azure fornecem comunicação de rede privada e segura. Com o acesso privado, as conexões com o servidor PostgreSQL ficam restritas à sua rede virtual. Para saber mais sobre isso, consulte [Acesso privado \(integração VNet\)](#).

Você pode implantar seu servidor flexível em uma rede virtual e em uma sub-rede durante a criação do servidor. Depois que o servidor flexível for implantado, você não poderá movê-lo para outra rede virtual, sub-rede ou para *acesso público (endereços IP permitidos)*.

Pré-requisitos

Para criar um servidor flexível em uma rede virtual, são necessários:

- Uma [rede virtual](#)

NOTE

A rede virtual e a sub-rede devem estar na mesma região e assinatura que o servidor flexível.

- [Delegar uma sub-rede para Microsoft. DBforPostgreSQL/flexibleServers](#). Essa delegação significa que somente os Servidores Flexíveis do Banco de Dados do Azure para PostgreSQL podem usar essa sub-rede. Nenhum outro tipo de recurso do Azure pode estar na sub-rede delegada.
- Adicione `Microsoft.Storage` ao ponto de extremidade de serviço para a sub-rede delegada a Servidores flexíveis. Isto é feito executando as seguintes etapas:
 1. Acesse sua página de rede virtual.
 2. Selecione a VNET na qual você está planejando implantar seu servidor flexível.
 3. Escolha a sub-rede que é delegada para o servidor flexível.
 4. Na tela de pull, em **Ponto de extremidade de serviço**, escolha `Microsoft.storage` na lista suspensa.
 5. Salve as alterações.

- Se você desejar configurar sua própria zona de DNS privado para usar com o servidor flexível, consulte a documentação de [visão geral de DNS privado](#) para obter mais detalhes.

Criar o Banco de Dados do Azure para PostgreSQL - Servidor Flexível em uma rede virtual já existente

1. Selecione **Criar um recurso** (+) no canto superior esquerdo do portal.
2. Selecione **Bancos de Dados > Banco de Dados do Azure para PostgreSQL**. Você também pode inserir **PostgreSQL** na caixa de pesquisa para localizar o serviço.
3. Selecione **Servidor flexível** como a opção de implantação.
4. Preencha o formulário **Noções básicas**.
5. Vá para a guia **Rede** para configurar o modo como você deseja se conectar ao servidor.
6. No **Método de conectividade**, selecione **Acesso privado (integração VNet)**. Vá até a **Rede virtual** e selecione a *rede virtual* já existente e a *Sub-rede* criada como parte dos pré-requisitos acima.
7. Em **Integração de DNS Privado**, por padrão, uma nova zona de DNS privado será criada usando o nome do servidor. Opcionalmente, você pode escolher a *assinatura* e a *zona de DNS privado* na lista suspensa.
8. Selecione **Examinar + criar** para examinar a configuração do servidor flexível.
9. Selecione **Criar** para provisionar o servidor. O provisionamento pode levar alguns minutos.

Flexible server (Preview)

...

Microsoft - preview

Network connectivity

You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

Connectivity method ⓘ

- Public access (allowed IP addresses)
 Private access (VNet Integration)



Connections from within the virtual network configured below will have access to this server. [Learn more](#)

Virtual network

Virtual networks are logically isolated from each other in Azure. Virtual network gives you a highly secure environment to run your PostgreSQL Flexible Server and other types of Azure resources

Subscription * ⓘ

Virtual network ⓘ

[Create virtual network](#)

Subnet * ⓘ

Private DNS integration

To connect privately with your private endpoint, you need a DNS record. We recommend that you integrate your private endpoint with a private DNS zone. You can also utilize your own DNS servers or create DNS records using the host files on your virtual machines. [Learn more](#)

Configuration Name	Subscription name	Private DNS zone
mytestserver-private...	<input type="text"/>	<input type="text"/> (New) mytestserver.private.postgres.datab...

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS). You can download the public SSL certificate from the above menu.
For information on TLS version and certificates, refer to connecting with TLS/SSL. [Learn more](#)

[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

NOTE

Depois que o servidor flexível for implantado na rede virtual e sub-rede, você não poderá movê-lo para Acesso público (endereços IP permitidos).

NOTE

Se desejar se conectar ao servidor flexível de um cliente provisionado em outra VNET, será necessário vincular a zona de DNS privado à VNET. Consulte a documentação [vinculando a rede virtual](#) para saber como fazer isso.

Próximas etapas

- [Criar e gerenciar a rede virtual do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a](#)

[CLI do Azure.](#)

- Saiba mais sobre a [rede no Banco de Dados do Azure para PostgreSQL - Servidor flexível](#)
- Saiba mais sobre a [rede virtual do Banco de Dados do Azure para PostgreSQL - Servidor flexível](#).

Criar e gerenciar redes virtuais do Banco de Dados do Azure para PostgreSQL - Servidor flexível usando a CLI do Azure

09/08/2021 • 4 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL dá suporte a dois tipos de métodos de conectividade de rede mutuamente exclusivos, para se conectar ao seu servidor flexível: As duas opções são:

- Acesso público (endereços IP permitidos)
- Acesso privado (Integração VNet)

Neste artigo, vamos enfocar a criação do servidor PostgreSQL com **Acesso privado (integração VNet)** usando a CLI do Azure. Com *Acesso privado (integração de VNet)*, é possível implantar o servidor flexível na [Rede Virtual do Azure](#). As Redes Virtuais do Azure fornecem comunicação de rede privada e segura. No acesso privado, as conexões com o servidor PostgreSQL são restritas apenas dentro da rede virtual. Para saber mais sobre isso, consulte [Acesso privado \(integração VNet\)](#).

No Banco de Dados do Azure para PostgreSQL - Servidor flexível, só é possível implantar o servidor em uma rede virtual e uma sub-rede durante a criação do servidor. Depois que o servidor flexível for implantado em uma rede virtual e sub-rede, não será possível movê-lo para outra rede virtual, sub-rede ou para *Acesso público (endereços IP permitidos)*.

Iniciar o Azure Cloud Shell

O [Azure Cloud Shell](#) é um shell gratuito e interativo que poderá ser usado para executar as etapas deste artigo. Ele tem ferramentas do Azure instaladas e configuradas para usar com sua conta.

Para abrir o Cloud Shell, basta selecionar **Experimentar** no canto superior direito de um bloco de código. Você também pode abrir o Cloud Shell em uma guia separada do navegador indo até <https://shell.azure.com/bash>. Selecione **Copiar** para copiar os blocos de código, cole-o no Cloud Shell e selecione **Enter** para executá-lo.

Caso prefira instalar e usar a CLI localmente, este início rápido exigirá a CLI do Azure versão 2.0 ou posterior. Execute `az --version` para encontrar a versão. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Pré-requisitos

Você precisará entrar na sua conta usando o comando `az login`. Observe a propriedade ID que se refere à **ID da Assinatura** da sua conta do Azure.

```
az login
```

Selecione a assinatura específica em sua conta usando o comando `az account set`. Anote o valor da ID da saída `az login` para usar como valor para o argumento **subscription** no comando. Se tiver várias assinaturas,

escolha a que for adequada para cobrança do recurso. Para obter todas as suas assinaturas, use `az account list`.

```
az account set --subscription <subscription id>
```

Criar um Banco de Dados PostgreSQL do Azure - Servidor flexível usando a CLI

É possível usar o comando `az postgres flexible-server` para criar o servidor flexível com *Acesso privado (Integração VNet)*. Esse comando usa o acesso privado (Integração VNet) como método de conectividade padrão. Uma rede virtual e uma sub-rede serão criadas para você se nenhuma for fornecida. Também é possível fornecer a rede virtual e a sub-rede já existentes usando a ID da sub-rede. Há várias opções para criar um servidor flexível usando a CLI, conforme mostrado nos exemplos abaixo.

IMPORTANT

O uso desse comando delegará a sub-rede para **Microsoft.DBforPostgreSQL/flexibleServers**. Essa delegação significa que somente os Servidores Flexíveis do Banco de Dados do Azure para PostgreSQL podem usar essa sub-rede. Nenhum outro tipo de recurso do Azure pode estar na sub-rede delegada.

Consulte a documentação de referência da CLI do Azure para obter uma lista completa de parâmetros configuráveis da CLI. Por exemplo, nos comandos abaixo, é possível, como opção, especificar o grupo de recursos.

- Criar um servidor flexível usando uma rede virtual padrão, uma sub-rede com o prefixo de endereço padrão

```
az postgres flexible-server create
```

- Crie um servidor flexível usando a rede virtual e a sub-rede já existentes. Se a rede virtual e a sub-rede fornecidas não existirem, a rede virtual e a sub-rede com o prefixo de endereço padrão serão criadas.

```
az postgres flexible-server create --vnet myVnet --subnet mySubnet
```

- Crie um servidor flexível usando a rede virtual já existente, a sub-rede e o usando a ID da sub-rede. A sub-rede fornecida não deve ter nenhum outro recurso implantado nela, sendo que essa sub-rede será delegada para **Microsoft.DBforPostgreSQL/flexibleServers**, se ainda não tiver sido delegada.

```
az postgres flexible-server create --subnet
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.Network/virtualNetworks/{VN
etName}/subnets/{SubnetName}
```

NOTE

A rede virtual e a sub-rede devem estar na mesma região e assinatura que o servidor flexível.

IMPORTANT

Os nomes incluindo `AzureFirewallSubnet`, `AzureFirewallManagementSubnet`, `AzureBastionSubnet` e `GatewaySubnet`, são nomes reservados no Azure. Não use-os como o nome da sub-rede.

- Crie um servidor flexível usando uma nova rede virtual, uma sub-rede com um prefixo de endereço não padrão

```
az postgres flexible-server create --vnet myVnet --address-prefixes 10.0.0.0/24 --subnet mySubnet --  
subnet-prefixes 10.0.0.0/24
```

Consulte a [documentação de referência](#) da CLI do Azure para obter a lista completa de parâmetros configuráveis da CLI.

IMPORTANT

Se você receber um erro `The parameter PrivateDnsZoneArguments is required, and must be provided by customer`, isso significa que pode estar executando uma versão mais antiga da CLI do Azure. [Atualize a CLI do Azure](#) e repita a operação.

Próximas etapas

- Saiba mais sobre a [rede no Banco de Dados do Azure para PostgreSQL - Servidor flexível](#).
- [Criar e gerenciar a rede virtual do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando o Portal do Azure](#).
- Saiba mais sobre a [rede virtual do Banco de Dados do Azure para PostgreSQL - Servidor flexível](#).

Criar e gerenciar regras de firewall para o Banco de Dados do Azure para PostgreSQL – Servidor flexível usando o portal do Azure

09/08/2021 • 5 minutes to read

IMPORTANT

O Banco de Dados do Azure para PostgreSQL – Servidor Flexível atualmente está em versão prévia pública.

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL dá suporte a dois tipos de métodos de conectividade de rede mutuamente exclusivos, para se conectar ao seu servidor flexível: As duas opções são:

- Acesso público (endereços IP permitidos)
- Acesso privado (Integração VNet)

Neste artigo, nos concentraremos na criação do servidor PostgreSQL com **Acesso público (endereços IP permitidos)** usando o portal do Azure e forneceremos uma visão geral do gerenciamento de regras de firewall após a criação de um servidor flexível. Com *acesso público (endereços IP permitidos)*, as conexões ao servidor PostgreSQL são restritas somente a endereços IP permitidos. Os endereços IP do cliente precisam ser permitidos nas regras de firewall. Para saber mais sobre isso, consulte [acesso público \(endereços IP permitidos\)](#). As regras de firewall podem ser definidas no momento da criação do servidor (recomendado), mas também podem ser adicionadas posteriormente. Neste artigo, forneceremos uma visão geral de como criar e gerenciar regras de firewall usando o acesso público (endereços IP permitidos).

Criar uma regra de firewall ao criar um servidor

1. Selecione **Criar um recurso** (+) no canto superior esquerdo do portal.
2. Selecione **Bancos de Dados > Banco de Dados do Azure para PostgreSQL**. Você também pode inserir **PostgreSQL** na caixa de pesquisa para localizar o serviço.
3. Selecione **Servidor flexível** como a opção de implantação.
4. Preencha o formulário **Noções básicas**.
5. Vá para a guia **Rede** para configurar o modo como você deseja se conectar ao servidor.
6. No **Método de conectividade**, selecione *Acesso público (endereços IP permitidos)*. Para criar as **Regras de firewall**, especifique o nome da Regra de firewall e o endereço IP único ou um intervalo de endereços. Se você quiser limitar a regra a um único endereço IP, digite o mesmo endereço no campo para endereço IP inicial e endereço IP final. Abrir o firewall permite que os administradores, usuários e aplicativos acessem os bancos de dados no servidor PostgreSQL para o qual eles têm credenciais válidas.

NOTE

O servidor flexível do Banco de Dados do Azure para PostgreSQL - o cria um firewall no nível do servidor. Ele impede que os aplicativos e as ferramentas externas se conectem ao servidor e aos bancos de dados no servidor, a menos que uma regra seja criada para abrir o firewall em endereços IP específicos.

7. Selecione **Examinar + criar** para examinar a configuração do servidor flexível.
8. Selecione **Criar** para provisionar o servidor. O provisionamento pode levar alguns minutos.

Criar uma regra de firewall após a criação do servidor

1. No [portal do Azure](#), selecione o Servidor flexível do Banco de Dados PostgreSQL do Azure no qual você deseja adicionar regras de firewall.
2. Na página servidor flexível, no título **Configurações**, clique em **Rede** para abrir a página Rede para o servidor flexível.
3. Clique em **Adicionar endereço IP do cliente atual** nas regras de firewall. Isso cria automaticamente uma regra de firewall com o endereço IP público do seu computador, como visto pelo sistema do Azure.
4. Verifique seu endereço IP antes de salvar a configuração. Em algumas situações, o endereço IP observado pelo Portal do Azure é diferente do endereço IP usado ao acessar a Internet e os servidores do Azure. Portanto, talvez seja necessário alterar o endereço IP inicial e o endereço IP final para que a regra funcione conforme o esperado.
Use um mecanismo de pesquisa ou outra ferramenta online para verificar seu próprio endereço IP. Por exemplo, pesquise "qual é meu IP".
5. Adicionar outros intervalos de endereço. Nas regras do firewall do Servidor flexível do Banco de Dados do Azure para PostgreSQL, você pode especificar um único endereço IP ou um intervalo de endereços. Se você quiser limitar a regra a um único endereço IP, digite o mesmo endereço no campo para endereço IP inicial e endereço IP final. Abrir o firewall permite que os administradores, usuários e aplicativos acessem os bancos de dados no servidor PostgreSQL para o qual eles têm credenciais válidas.
6. Clique em **Salvar** na barra de ferramentas para salvar essa regra de firewall. Aguarde a confirmação de que a atualização das regras de firewall foi bem-sucedida.

Conexão pelo Azure

Talvez você queira habilitar recursos ou aplicativos implantados no Azure para se conectar ao seu servidor flexível. Isso inclui aplicativos Web hospedados no Serviço de Aplicativo do Azure, em execução em uma VM do Azure, em um gateway de gerenciamento de dados Azure Data Factory e muito mais.

Quando um aplicativo no Azure tenta se conectar ao seu servidor, o firewall verifica se as conexões do Azure são permitidas. Você pode habilitar essa configuração selecionando a opção **Permitir acesso público dos serviços e recursos do Azure no Azure para este servidor** no portal, na guia **Rede** e clicando em **Salvar**.

Os recursos não precisam estar na mesma rede virtual (VNet) ou grupo de recursos para a regra de firewall habilitar essas conexões. Se a tentativa de conexão não for permitida, a solicitação não alcançará o Servidor flexível do Banco de Dados do Azure para PostgreSQL.

IMPORTANT

Esta opção configura o firewall para permitir todas as conexões do Azure, incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

É recomendável escolher o **Acesso privado (integração VNet)** para acessar com segurança o servidor flexível.

Gerenciar as regras de firewall existentes no portal do Azure

Repita as etapas a seguir para gerenciar as regras de firewall.

- Para adicionar o computador atual, clique em + **Adicionar endereço IP do cliente atual** nas regras de firewall. Clique em **Salvar** para salvar as alterações.
- Para adicionar mais endereços IP, digite o Nome da Regra, o Endereço IP Inicial e o Endereço IP Final. Clique

em **Salvar** para salvar as alterações.

- Para modificar uma regra existente, clique em qualquer um dos campos na regra e modifique. Clique em **Salvar** para salvar as alterações.
- Para excluir uma regra existente, clique nas reticências [...] e clique em **Excluir** para remover a regra. Clique em **Salvar** para salvar as alterações.

Próximas etapas

- Saiba mais sobre a [rede no Banco de Dados do Azure para PostgreSQL - Servidor flexível](#)
- Reconheça mais sobre as[Funções do firewall do Servidor Flexível do Banco de Dados do Azure para PostgreSQL](#)
- [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando a CLI do Azure.](#)

Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a CLI do Azure/

21/05/2021 • 7 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL dá suporte a dois tipos de métodos de conectividade de rede mutuamente exclusivos, para se conectar ao seu servidor flexível: As duas opções são:

- Acesso público (endereços IP permitidos)
- Acesso privado (Integração VNet)

Neste artigo, nos concentraremos na criação do servidor PostgreSQL com **acesso público (endereços IP permitidos)** usando CLI do Azure e fornecerão uma visão geral sobre os comandos do CLI do Azure que podem ser usados para criar, atualizar, excluir, listar e mostrar regras de firewall após a criação do servidor. Com **acesso público (endereços IP permitidos)**, as conexões ao servidor PostgreSQL são restritas somente a endereços IP permitidos. Os endereços IP do cliente precisam ser permitidos nas regras de firewall. Para saber mais sobre isso, veja o [acesso público \(endereços IP permitidos\)](#). As funções de firewall podem ser definidas no momento da criação do servidor (recomendado), mas também podem ser adicionadas posteriormente.

Iniciar o Azure Cloud Shell

O [Azure Cloud Shell](#) é um shell gratuito e interativo que poderá ser usado para executar as etapas deste artigo. Ele tem ferramentas do Azure instaladas e configuradas para usar com sua conta.

Para abrir o Cloud Shell, basta selecionar **Experimentar** no canto superior direito de um bloco de código. Você também pode abrir o Cloud Shell em uma guia separada do navegador indo até <https://shell.azure.com/bash>. Selecione **Copiar** para copiar os blocos de código, cole-o no Cloud Shell e selecione **Enter** para executá-lo.

Caso prefira instalar e usar a CLI localmente, este início rápido exigirá a CLI do Azure versão 2.0 ou posterior. Execute `az --version` para encontrar a versão. Se você precisa instalar ou atualizar, consulte [Instalar a CLI do Azure](#).

Pré-requisitos

Você precisará entrar na sua conta usando o comando `az login`. Observe a propriedade **ID** que se refere à **ID da Assinatura** da sua conta do Azure.

```
az login
```

Selecione a assinatura específica em sua conta usando o comando `az account set`. Anote o valor da ID da saída `az login` para usar como valor para o argumento **subscription** no comando. Se tiver várias assinaturas, escolha a que for adequada para cobrança do recurso. Para obter todas as suas assinaturas, use `az account list`.

```
az account set --subscription <subscription id>
```

Crie a regra de firewall durante a criação de servidor flexível usando o CLI do Azure

Você pode usar o `az postgres flexible-server --public access` comando para criar o servidor flexível com *acesso público* (*endereços IP permitidos*) e configurar as regras de firewall durante a criação de um servidor flexível. Você pode usar a opção `--Public-Access` para fornecer os endereços IP permitidos que serão capazes de se conectar ao servidor. Você pode fornecer um simples ou intervalo de endereços IP a serem incluídos na lista de IPs permitidos. O intervalo de endereços IP deve ser separado por um traço e não conter espaços. Há diversas opções para criar um servidor flexível usando a CLI, conforme mostrado no exemplo seguinte.

Consulte a documentação de referência da CLI do Azure para obter uma lista completa de parâmetros configuráveis da CLI. Por exemplo, nos comandos abaixo, é possível, como opção, especificar o grupo de recursos.

- Criar um servidor flexível de acesso público e adicione o endereço IP do cliente para ter acesso ao servidor

```
az postgres flexible-server create --public-access <my_client_ip>
```

- Criar um servidor flexível de acesso público e adicionar o intervalo de endereços IP para ter acesso a esse servidor

```
az postgres flexible-server create --public-access <start_ip_address-end_ip_address>
```

- Criar um servidor flexível de acesso público e permitir que os aplicativos de endereços IP do Azure se conectem ao seu servidor flexível

```
az postgres flexible-server create --public-access 0.0.0.0
```

IMPORTANT

Esta opção configura o firewall para permitir todas as conexões do Azure incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

- Ao criar um servidor flexível com acesso público e permitir todos os endereços IP

```
az postgres flexible-server create --public-access all
```

NOTE

O comando acima criará uma função de firewall com endereço IP inicial = 0.0.0.0, endereço IP final = 255.255.255.255 e nenhum endereço IP será bloqueado. Nenhum hospedeiro na Internet pode acessar este servidor. É extremamente recomendado usar essa regra apenas temporariamente e somente em servidores de teste que não contenham dados confidenciais.

- Criar um servidor flexível de acesso público e sem endereço IP

```
az postgres flexible-server create --public-access none
```

NOTE

Não recomendamos criar um servidor sem nenhuma função do firewall. Se você não adicionar nenhuma função do firewall, nenhum cliente poderá se conectar ao servidor.

Criar e gerenciar a função do firewall após a criação do servidor

O comando **az postgres flexible-server firewall-rule** é usado na CLI do Azure para criar, excluir, listar, exibir e atualizar funções do firewall.

Comandos:

- **criar**: crie uma função do firewall no servidor flexível do MySQL do Azure.
- **listar**: liste as funções do firewall de servidor flexível do MySQL do Azure.
- **atualizar**: Atualize uma função do firewall do servidor flexível do MySQL do Azure.
- **show**: Mostre os detalhes de uma regra de firewall de servidor flexível do Azure.
- **delete**: Exclua uma regra de firewall de servidor flexível MySQL do Azure.

Consulte a documentação de referência da CLI do Azure para obter uma lista completa de parâmetros configuráveis da CLI. Por exemplo, nos comandos abaixo, é possível, como opção, especificar o grupo de recursos.

Criar uma regra de firewall

Use o `az postgres flexible-server firewall-rule create` comando para criar uma nova regra de firewall no servidor de rede. Para permitir o acesso a um intervalo de endereço IP, forneça como endereço IP Inicial e endereço IP Final, como neste exemplo.

```
az postgres flexible-server firewall-rule create --name mydemoserver --start-ip-address 13.83.152.0 --end-ip-address 13.83.152.15
```

Para permitir o acesso de um endereço IP único, apenas forneça o endereço IP, como neste exemplo.

```
az postgres flexible-server firewall-rule create --name mydemoserver --start-ip-address 1.1.1.1
```

Para permitir que aplicativos dos endereços IP do Azure conectem-se ao seu Banco de Dados do Azure para servidor flexível do MySQL, forneça o endereço IP 0.0.0.0 como IP Inicial e IP Final, como neste exemplo.

```
az postgres flexible-server firewall-rule create --name mydemoserver --start-ip-address 0.0.0.0
```

IMPORTANT

Esta opção configura o firewall para permitir todas as conexões do Azure incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

Após o êxito, cada saída de comando listará os detalhes da regra de firewall que você criou, no formato JSON (por padrão). Se houver uma falha, a saída mostrará o texto da mensagem de erro em vez disso.

Listar regras de firewall

Use o `az postgres flexible-server firewall-rule list` comando para listar as funções do firewall do servidor existentes no servidor. Observe que o atributo de nome do servidor é especificado na opção `--opção` interruptor.

```
az postgres flexible-server firewall-rule list --name mydemoserver
```

A saída listará as regras, se houver, no formato JSON (por padrão). É possível usar a opção `--output table` para gerar os resultados em um formato de tabela mais legível.

```
az postgres flexible-server firewall-rule list --name mydemoserver --output table
```

Atualizar uma regra de firewall

Use o `az postgres flexible-server firewall-rule update` comando para atualizar uma função do firewall existente no servidor. Forneça o nome da regra de firewall existente como entrada, bem como os atributos de endereço IP inicial e IP final a serem atualizados.

```
az postgres flexible-server firewall-rule update --name mydemoserver --rule-name FirewallRule1 --start-ip-address 13.83.152.0 --end-ip-address 13.83.152.1
```

Após o êxito, a saída do comando listará os detalhes da regra do firewall atualizada, em formato JSON (por padrão). Se houver uma falha, a saída mostrará o texto da mensagem de erro em vez disso.

NOTE

Se a regra de firewall não existir, ela será criada pelo comando de atualização.

Mostrar detalhes da regra de firewall

Use o `az postgres flexible-server firewall-rule show` comando para mostrar os detalhes da função do firewall existente do servidor. Forneça o nome da regra de firewall existente como entrada.

```
az postgres flexible-server firewall-rule show --name mydemoserver --rule-name FirewallRule1
```

Após o êxito, a saída do comando listará os detalhes da regra de firewall especificado, em formato JSON (por padrão). Se houver uma falha, a saída mostrará o texto da mensagem de erro em vez disso.

Excluir uma regra de firewall

Use o `az postgres flexible-server firewall-rule delete` comando para excluir uma regra de firewall existente no servidor. Forneça o nome da regra de firewall existente.

```
az postgres flexible-server firewall-rule delete --name mydemoserver --rule-name FirewallRule1
```

Após o êxito, não haverá saída. Em caso de falha, o texto da mensagem de erro será exibido.

Próximas etapas

- Saiba mais sobre a [rede no Banco de Dados do Azure para PostgreSQL - Servidor flexível](#)
- Reconheça mais sobre as [Funções do firewall do Servidor Flexível do Banco de Dados do Azure para PostgreSQL](#)

- Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando o Portal do Azure.

Conectividade criptografada usando o TLS (Transport Layer Security) no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

21/05/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

O Banco de Dados do Azure para PostgreSQL Servidor Flexível dá suporte à conexão dos aplicativos clientes ao serviço do PostgreSQL usando TLS (Protocolo TLS), anteriormente conhecido como SSL (Protocolo SSL). O TLS é um protocolo padrão do setor que garante conexões de rede criptografadas entre o servidor de banco de dados e os aplicativos cliente, permitindo que você atenda aos requisitos de conformidade.

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL dá suporte a conexões criptografadas usando TLS 1.2 (protocolo TLS) e todas as conexões de entrada com TLS 1.0 e TLS 1.1 serão negadas. Para todos os servidores flexíveis, a imposição de conexões TLS está habilitada e você não pode desabilitar o TLS/SSL para se conectar ao servidor flexível.

Aplicativos que exigem a verificação de certificado para conectividade TLS/SSL

Em alguns casos, os aplicativos exigem um arquivo de certificado local gerado de um arquivo de certificado de uma Autoridade de Certificação (CA) confiável para se conectar com segurança. Banco de dados do Azure para PostgreSQL-o servidor flexível usa a *autoridade de certificação raiz global do DigiCert*. Baixe esse certificado necessário para se comunicar por SSL da [AC raiz global do DigiCert](#) e salve o arquivo de certificado em seu local preferido. Por exemplo, este tutorial usa `c:\ssl`.

Conecte-se usando psql

Caso tenha criado um servidor flexível com *acesso privado (Integração VNet)*, será necessário se conectar ao servidor de um recurso na mesma VNet do servidor. Crie uma máquina virtual e adicione-a à VNET criada com o servidor flexível.

Caso tenha criado um servidor flexível com *acesso público (endereços IP permitidos)*, será possível adicionar seu endereço IP local à lista de regras de firewall no servidor.

O exemplo a seguir mostra como se conectar ao servidor usando a interface de linha de comando do PSQL. Use a configuração de cadeia de conexão `sslmode=verify-full` para impor a verificação de certificado TLS/SSL. Passe o caminho do arquivo de certificado local para o parâmetro `sslrootcert`.

```
psql "sslmode=verify-full sslrootcert=c:\ssl\DigicertGlobalRootCA.crt.pem host=mydemoserver.postgres.database.azure.com dbname=postgres user=myadmin"
```

NOTE

Confirme se o valor passado para `sslrootcert` corresponde ao caminho do arquivo para o certificado que você salvou.

Verificar se o seu aplicativo ou sua estrutura oferece suporte a conexões TLS

Algumas estruturas do aplicativos que usam o PostgreSQL para serviços de banco de dados não habilitam o protocolo TSL por padrão durante a instalação. O seu servidor PostgreSQL impõe conexões TLS, mas o aplicativo não está configurado para o TLS, o aplicativo pode falhar ao se conectar ao seu servidor de banco de dados. Confira a documentação de seu aplicativo para saber como habilitar conexões TLS.

Próximas etapas

- [Criar e gerenciar a rede virtual do Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a CLI do Azure.](#)
- Saiba mais sobre a [rede no Banco de Dados do Azure para PostgreSQL - Servidor flexível](#)
- Reconheça mais sobre as[Funções do firewall do Servidor Flexível do Banco de Dados do Azure para PostgreSQL](#)

Configurar os parâmetros do servidor no Banco de Dados do Azure para PostgreSQL – Servidor Flexível por meio do portal do Azure

09/08/2021 • 2 minutes to read

Você pode listar, exibir e atualizar os parâmetros de configuração para um servidor flexível do Banco de Dados do Azure para PostgreSQL usando o portal do Azure.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- [Servidor flexível do Banco de Dados do Azure para PostgreSQL](#)

Exibir e editar parâmetros

1. Abra o [Portal do Azure](#).
2. Selecione o seu servidor flexível.
3. Sob a seção **configurações**, selecione **parâmetros de servidor**. A página mostra uma lista de parâmetros, valores e descrições.

The screenshot shows the 'Server parameters' page for the 'demoserver20' PostgreSQL flexible server. The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Settings (Compute + storage, Networking, Connection strings), and Monitoring (Alerts, Metrics, Diagnostic settings). The 'Server parameters' section is currently selected. The main area displays a table of parameters with columns for Parameter name, Value, Parameter type, and Description. Each parameter row includes a small info icon and three dots for more details. The 'application_name' parameter is set to 'Dynamic' with a value of 'demoserver20'. Other parameters shown include 'array_nulls' (ON), 'autovacuum' (ON), 'autovacuum_analyze_scale_factor' (0.1), 'autovacuum_analyze_threshold' (50), 'autovacuum_freeze_max_age' (20000000), 'autovacuum_max_workers' (1), 'autovacuum_multixact_freeze_max...', 'autovacuum_naptime' (60), 'autovacuum_vacuum_cost_delay' (2), 'autovacuum_vacuum_cost_limit' (-1), 'autovacuum_vacuum_scale_factor' (0.2), and 'autovacuum_vacuum_threshold' (50).

Parameter name	Value	Parameter type	Description
application_name		Dynamic	Sets the application name to be reported in statistic...
array_nulls	ON OFF	Dynamic	Enables input of NULL (case insensitive) to be considered...
autovacuum	ON OFF	Dynamic	Controls whether the server should run the autova...
autovacuum_analyze_scale_factor	0.1	Dynamic	Specifies a fraction of the table size to add to auto...
autovacuum_analyze_threshold	50	Dynamic	Sets the minimum number of inserted, updated or...
autovacuum_freeze_max_age	20000000	Static	Maximum age (in transactions) before triggering a...
autovacuum_max_workers	1	Static	Sets the maximum number of simultaneously running...
autovacuum_multixact_freeze_max...	40000000	Static	Maximum age (in multixact) before triggering aut...
autovacuum_naptime	60	second: Dynamic	Sets minimum delay between autovacuum runs or...
autovacuum_vacuum_cost_delay	2	millisec Dynamic	Sets cost delay value (milliseconds) that will be used...
autovacuum_vacuum_cost_limit	-1	Dynamic	Sets cost limit value that will be used in automatic...
autovacuum_vacuum_scale_factor	0.2	Dynamic	Specifies a fraction of the table size to add to auto...
autovacuum_vacuum_threshold	50	Dynamic	SPECIFIES THE MINIMUM NUMBER OF UPDATED OR DELETED...

4. Selecione o botão **suspensão** para ver os possíveis valores para parâmetros de tipo enumerado como `client_min_messages`.

demoserver20 | Server parameters

Azure Database for PostgreSQL flexible server

Search (Ctrl+ /) Save Discard Reset all to default Feedback

backend_flush_after	256	8kiloby	Dynamic	Number of pages after which previously performe...	...
backslash_quote	SAFE_ENCODING	Dynamic	Dynamic	Sets whether \" is allowed in string literals.	...
bgwriter_delay	50	millisec	Dynamic	Specifies the delay between activity rounds for the...	...
bgwriter_flush_after	DEBUG5	8kiloby	Dynamic	Number of pages after which previously performe...	...
bgwriter_lru_maxpages	DEBUG4	Dynamic	Dynamic	In each round, no more than this many buffers will...	...
bgwriter_lru_multiplier	DEBUG3	Dynamic	Dynamic	The average recent need of buffers is multiplied b...	...
bytea_output	DEBUG2	Dynamic	Dynamic	Sets the output format for values of type bytea. Va...	...
check_function_bodies	DEBUG1	Dynamic	Dynamic	Checks function bodies during CREATE FUNCTION.	...
checkpoint_completion_target	LOG	Dynamic	Dynamic	Specifies the target of checkpoint completion, as a...	...
checkpoint_timeout	NOTICE	second	Dynamic	Maximum time between automatic WAL checkpoi...	...
checkpoint_warning	WARNING	second	Dynamic	Writes a warning message if checkpoints caused b...	...
client_encoding	ERROR	Dynamic	Dynamic	Sets the client-side encoding (character set). The d...	...
client_min_messages	NOTICE	Dynamic	Dynamic	Controls the message levels that are sent to the cli...	...
commit_delay	0	microse	Dynamic	Sets the delay in microseconds between transactio...	...
commit_siblings	5	Dynamic	Dynamic	Sets the minimum concurrent open transactions b...	...
connection_throttle.bucket_limit	2000	Dynamic	Dynamic	Max login tokens per bucket.	...
connection_throttle.enable	ON OFF	Dynamic	Dynamic	Enables temporary connection throttling per IP for...	...

5. Selecione ou passe o mouse sobre o botão **i** (informações) para ver o intervalo de valores possíveis para parâmetros numéricos como `cpu_index_tuple_cost`.

Home > Azure Database for PostgreSQL servers > demoserver20

demoserver20 | Server parameters

Azure Database for PostgreSQL flexible server

Search (Ctrl+ /) Save Discard Reset all to default Feedback

connection_throttle.enable	ON OFF	Dynamic	Dynamic	Enables temporary connection throttling per IP for...	...
connection_throttle.factor_bias	0.8	Dynamic	Dynamic	The factor bias for calculating number of tokens f...	...
connection_throttle.hash_entries_...	500	Dynamic	Dynamic	Max number of entries in the login failures hash ta...	...
connection_throttle.reset_time	120	second	Dynamic	Time between resetting the login bucket.	...
connection_throttle.restore_factor	2	Dynamic	Dynamic	Factor to increase number of tokens by for IPs wit...	...
connection_throttle.update_time	20	second	Dynamic	Time between updating the login bucket.	...
constraint_exclusion	PARTI	Allowed value should be: 0-1.79769e+308	Dynamic	Controls the query planner's use of table constrain...	...
cpu_index_tuple_cost	0.005	Dynamic	Dynamic	Sets the planner's estimate of the cost of processi...	...
cpu_operator_cost	0.0025	Dynamic	Dynamic	Sets the planner's estimate of the cost of processi...	...
cpu_tuple_cost	0.01	Dynamic	Dynamic	Sets the planner's estimate of the cost of processi...	...
cron.database_name	postgres	Static	Static	Sets the database in which pg_cron metadata is ke...	...
cron.log_run	ON OFF	Static	Static	Log all jobs runs into the job_run_details table.	...
cron.log_statement	ON OFF	Static	Static	Log all cron statements prior to execution.	...
cron.max_running_jobs	32	Static	Static	Sets the maximum number of jobs that can run co...	...
cursor_tuple_fraction	0.1	Dynamic	Dynamic	Sets the planner's estimate of the fraction of a cur...	...
DateStyle	ISO, MDY	Dynamic	Dynamic	Sets the display format for date and time values.	...
deadlock_timeout	1000	millisec	Dynamic	Sets the amount of time, in milliseconds, to wait o...	...

6. Se necessário, use a **caixa de pesquisa** para restringir rapidamente a um parâmetro específico. A busca está no nome e na descrição dos parâmetros.

demoserver20 | Server parameters

Azure Database for PostgreSQL flexible server

Search (Ctrl+ /) Save Discard Reset all to default Feedback

This list shows modifiable server parameters. Please click on the info box to get more details about a particular parameter, like the allowed values and data type. [Learn more](#)

max_running_jobs

Parameter name	Value	Parameter type	Description
cron.max_running_jobs	32	Static	Sets the maximum number of jobs that can run con...

Overview Activity log Access control (IAM) Tags Settings Compute + storage Networking Connection strings Server parameters Maintenance High availability Locks Monitoring Alerts Metrics Diagnostic settings

7. Altere os valores de parâmetro que você deseja ajustar. Todas as alterações feitas nessa sessão são realçadas em roxo. Depois de alterar os valores, você pode selecionar **Salvar**. Ou então, você pode **Descartar** suas alterações.

Home > demoserver20

demoserver20 | Server parameters

Azure Database for PostgreSQL flexible server

Save Discard Reset all to default Feedback

There are 1 unsaved parameter changes. Please save to apply these updates.

Search to filter items...

Parameter name	Value	Parameter type	Description	
application_name		Dynamic	Sets the application name to be reported in statist...	
array_nulls	ON OFF	Dynamic	Enables input of NULL (case insensitive) to be con...	
autovacuum	ON OFF	Dynamic	Controls whether the server should run the autova...	
autovacuum_analyze_scale_factor	0.1	Dynamic	Specifies a fraction of the table size to add to auto...	
autovacuum_analyze_threshold	50	Dynamic	Sets the minimum number of inserted, updated or...	
autovacuum_freeze_max_age	200000000	Static	Maximum age (in transactions) before triggering a...	
autovacuum_max_workers	1	Static	Sets the maximum number of simultaneously runn...	
autovacuum_multixact_freeze_max...	400000000	Static	Maximum age (in multixact) before triggering aut...	
autovacuum_naptime	120	second	Dynamic	Sets minimum delay between autovacuum runs o...
autovacuum_vacuum_cost_delay	2	millisec	Dynamic	Sets cost delay value (milliseconds) that will be us...
autovacuum_vacuum_cost_limit	-1	Dynamic	Sets cost limit value that will be used in automatic ...	
autovacuum_vacuum_scale_factor	0.2	Dynamic	Specifies a fraction of the table size to add to auto...	
autovacuum_vacuum_threshold	50	Dynamic	Specifies the minimum number of updated or dele...	
autovacuum_work_mem	-1	kilobyte	Dynamic	Sets the maximum memory to be used by each au...

Activity log Access control (IAM) Tags Settings Compute + storage Networking Connection strings Server parameters Maintenance High availability Locks Monitoring Alerts Metrics Diagnostic settings Logs

8. Se você tiver salvo os novos valores para os parâmetros, você sempre pode reverter tudo o que fazer com os valores padrão selecionando **Redefinir tudo para o padrão**.

demoserver20 | Server parameters ...

Azure Database for PostgreSQL flexible server

Setting	Value	Type	Description	Actions	
array_nulls	<input checked="" type="radio"/> ON <input type="radio"/> OFF	Dynamic	Enables input of NULL (case insensitive) to be con...	...	
autovacuum	<input checked="" type="radio"/> ON <input type="radio"/> OFF	Dynamic	Controls whether the server should run the autova...	...	
autovacuum_analyze_scale_factor	0.1	Dynamic	Specifies a fraction of the table size to add to auto...	...	
autovacuum_analyze_threshold	50	Dynamic	Sets the minimum number of inserted, updated or...	...	
autovacuum_freeze_max_age	20000000	Static	Maximum age (in transactions) before triggering a...	...	
autovacuum_max_workers	1	Static	Sets the maximum number of simultaneously runn...	...	
autovacuum_multixact_freeze_max...	40000000	Static	Maximum age (in multixact) before triggering aut...	...	
autovacuum_naptime	128	second	Dynamic	Sets minimum delay between autovacuum runs o...	...
autovacuum_vacuum_cost_delay	2	millisec	Dynamic	Sets cost delay value (milliseconds) that will be us...	...
autovacuum_vacuum_cost_limit	-1	Dynamic	Sets cost limit value that will be used in automatic	
autovacuum_vacuum_scale_factor	0.2	Dynamic	Specifies a fraction of the table size to add to auto...	...	
autovacuum_vacuum_threshold	50	Dynamic	Specifies the minimum number of updated or dele...	...	
autovacuum_work_mem	-1	kilobyte	Dynamic	Sets the maximum memory to be used by each au...	...
azure.accepted_password_auth_me...	MD5	Dynamic	Accepted password authentication method	...	
azure.enable_temp_tablespaces_on...	<input checked="" type="radio"/> ON <input type="radio"/> OFF	Dynamic	Create temp tablespace on ephemeral disk	...	
backend_flush_after	256	8kiloby	Dynamic	Number of pages after which previously performe...	...
backslash_quote	SAFE_ENCODING	Dynamic	Sets whether "\\" is allowed in string literals.	...	

Próximas etapas

Saiba mais:

- [Visão geral dos parâmetros do servidor no Banco de Dados do Azure para PostgreSQL](#)
- [Configurando parâmetros usando a CLI do Azure](#)

Personalizar parâmetros de servidor para o Banco de Dados do Azure para PostgreSQL – Servidor Flexível usando a CLI do Azure

21/05/2021 • 2 minutes to read

Você pode listar, exibir e atualizar os parâmetros de configuração de um servidor PostgreSQL do Azure usando a CLI (Interface de Linha de Comando) do Azure. Um subconjunto de parâmetros de mecanismo é exposto no nível do servidor e pode ser modificado.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Criar um banco de dados e um servidor de Banco de Dados do Azure para PostgreSQL seguindo [Criar um Banco de Dados do Azure para o servidor PostgreSQL](#)
- Instalar a interface de linha de comando da [CLI do Azure](#) no computador ou usar o [Azure Cloud Shell](#) no portal do Azure usando seu navegador.

Listar os parâmetros de servidor para um servidor flexível

Para listar todos os parâmetros modificáveis em um servidor e seus valores, execute o comando [az postgres flexible-server parameter list](#).

É possível listar os parâmetros do servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`.

```
az postgres flexible-server parameter list --resource-group myresourcegroup --server-name mydemoserver
```

Mostrar os detalhes do parâmetro do servidor

Para mostrar os detalhes de um parâmetro específico para um servidor, execute o comando [az postgres flexible-server parameter show](#).

Este exemplo mostra detalhes do parâmetro do servidor `log_min_messages` para o servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`.

```
az postgres flexible-server parameter show --name log_min_messages --resource-group myresourcegroup --server-name mydemoserver
```

Modificar o valor do parâmetro do servidor

Você pode modificar o valor de um determinado parâmetro que atualiza o valor da configuração subjacente para o mecanismo do servidor PostgreSQL. Para atualizar o parâmetro, use o comando [az postgres flexible-server parameter set](#).

Para atualizar o parâmetro `log_min_messages` do servidor `mydemoserver.postgres.database.azure.com` no grupo de recursos `myresourcegroup`.

```
az postgres flexible-server parameter set --name log_min_messages --value INFO --resource-group myresourcegroup --server-name mydemoserver
```

Se você quiser redefinir o valor de um parâmetro, omita o `--value` do parâmetro opcional e o serviço vai aplicar o valor padrão. No exemplo acima, ele teria a seguinte aparência:

```
az postgres flexible-server parameter set --name log_min_messages --resource-group myresourcegroup --server-name mydemoserver
```

Esse comando redefine o parâmetro `log_min_messages` para o valor padrão `WARNING`. Para obter mais informações sobre os parâmetros do servidor e os valores permitidos, confira a documentação do PostgreSQL em [Definição de parâmetros](#).

Próximas etapas

- Para configurar e acessar os logs de servidor, confira [Logs de servidor no Banco de Dados do Azure para PostgreSQL](#)

Operações de dimensionamento no servidor flexível

21/05/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo apresenta as etapas para realizar operações de dimensionamento para computação e armazenamento. Você poderá alterar as suas camadas de computação entre SKUs com capacidade de intermitência, de uso geral e otimizadas para memória, inclusive escolhendo o número de vCores adequados para executar o seu aplicativo. Você também pode escalar verticalmente o armazenamento. As IOPS esperadas são exibidas com base na camada de computação, nos vCores e na capacidade de armazenamento. A estimativa de custos também é exibida com base na sua seleção.

IMPORTANT

Não é possível reduzir verticalmente o armazenamento.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Você deve ter um Banco de Dados do Azure para PostgreSQL – servidor flexível. O mesmo procedimento também é aplicável para o servidor flexível configurado com redundância de zona.

IMPORTANT

Quando ele é configurado com alta disponibilidade, você não pode escolher a SKU com capacidade de intermitência. Durante a operação de dimensionamento, primeiro o servidor em espera é dimensionado para o tamanho desejado, o servidor primário faz failover e o primário é dimensionado.

Dimensionando o tamanho e a camada de computação

Siga as próximas etapas para escolher a camada de computação.

1. No [portal do Azure](#), escolha o servidor flexível do qual você deseja restaurar o backup.
2. Clique em **Computação+armazenamento**.
3. É exibida uma página com as configurações atuais.

Search (Ctrl+ /) <> Compute + storage

Overview

Activity log

Access control (IAM)

Tags

Settings

Compute + storage

Networking

Connection strings

Server parameters

Maintenance Window

High availability

Locks

Compute tier

Compute resources are pre-allocated and billed per minute based on vCores configured.

Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously

General Purpose (2-64 vCores) - Balanced configuration for most common workloads

Memory Optimized (2-64 vCores) - Best for workloads that require a high memory to CPU ratio

Compute size

Standard_D4s_v3 (4 vCores, 16 GB memory)

Storage size (in GiB)

512 1024 2048 4096 8192 16384

Storage cannot be scaled down

UP TO 2300 AVAILABLE IOPS

4. Para a classe de computação, você pode escolher entre camadas com capacidade de intermitência, de uso geral e otimizadas para memória.

Compute + storage

Compute resources are pre-allocated and billed per minute based on vCores configured.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (2-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (2-64 vCores) - Best for workloads that require a high memory to CPU ratio

5. Se você concordar com os tamanhos de memória e os vCores padrão, poderá ignorar a próxima etapa.

6. Se quiser alterar o número de vCores, clique no menu suspenso **Tamanho da computação** e, na lista, clique no número desejado de vCores/memória.

- Camada de computação com capacidade de intermitência:

Compute + storage

Compute resources are pre-allocated and billed per minute based on vCores configured.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (2-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (2-64 vCores) - Best for workloads that require a high memory to CPU ratio

Compute size

Standard_B1ms (1 vCore, 2 GB memory)

Standard_B1ms (1 vCore, 2 GB memory)

Standard_B2s (2 vCores, 4 GB memory)

- Camada de computação de uso geral:

Compute + storage

Compute resources are pre-allocated and billed per minute based on vCores configured.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (2-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (2-64 vCores) - Best for workloads that require a high memory to CPU ratio

Compute size

- Standard_D4s_v3 (4 vCores, 16 GB memory)
- Standard_D2s_v3 (2 vCores, 8 GB memory)
- Standard_D4s_v3 (4 vCores, 16 GB memory)
- Standard_D8s_v3 (8 vCores, 32 GB memory)
- Standard_D16s_v3 (16 vCores, 64 GB memory)
- Standard_D32s_v3 (32 vCores, 128 GB memory)
- Standard_D64s_v3 (64 vCores, 256 GB memory)

- Camada de computação otimizada para memória:

Compute + storage

Compute resources are pre-allocated and billed per minute based on vCores configured.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (2-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (2-64 vCores) - Best for workloads that require a high memory to CPU ratio

Compute size

- Standard_E2s_v3 (2 vCores, 16 GB memory)
- Standard_E2s_v3 (2 vCores, 16 GB memory)
- Standard_E4s_v3 (4 vCores, 32 GB memory)
- Standard_E8s_v3 (8 vCores, 64 GB memory)
- Standard_E16s_v3 (16 vCores, 128 GB memory)
- Standard_E32s_v3 (32 vCores, 256 GB memory)
- Standard_E64s_v3 (64 vCores, 432 GB memory)

7. Clique em **Salvar**.

8. Você verá uma mensagem de confirmação. Clique em **OK** se quiser continuar.

9. Você verá uma notificação sobre a operação de dimensionamento em andamento.

Dimensionando o tamanho do armazenamento

Siga estas etapas para aumentar o tamanho do armazenamento.

1. No [portal do Azure](#), escolha o servidor flexível cujo tamanho de armazenamento você deseja aumentar.
2. Clique em **Computação+armazenamento**.
3. É exibida uma página com as configurações atuais.

Search (Ctrl+ /) <> Compute + storage

Overview Compute resources are pre-allocated and billed per minute based on vCores configured.

Activity log Compute tier

Access control (IAM)

Tags

Settings

Compute + storage

Networking

Connection strings

Server parameters

Maintenance Window

High availability

Locks

Compute size Standard_D4s_v3 (4 vCores, 16 GB memory)

Storage size (in GiB) 512 1024 2048 4096 8192 16384 512

⚠ Storage cannot be scaled down

UP TO 2300 AVAILABLE IOPS

The screenshot shows the 'Compute + storage' settings page in the Azure portal. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Settings, and Compute + storage (which is selected and highlighted with a red border). Under Compute + storage, there are links for Networking, Connection strings, Server parameters, Maintenance Window, High availability, and Locks. The main area has sections for Compute size (set to Standard_D4s_v3) and Storage size (in GiB). A slider is shown for storage size, with values from 32 to 16384 GiB, currently set at 512. A note below the slider says 'Storage cannot be scaled down'. To the right of the slider, it says 'UP TO 2300 AVAILABLE IOPS'.

4. O campo **Tamanho do armazenamento em GiB** com uma barra deslizante é exibido com o tamanho atual.

5. Deslize a barra até o tamanho desejado. É exibido o número correspondente de IOPS. A IOPS depende do tamanho e da camada de computação. As informações de custo também são exibidas.



6. Se você concordar com o tamanho do armazenamento, clique em **Salvar**.

7. Você verá uma mensagem de confirmação. Clique em **OK** se quiser continuar.

8. Você verá uma notificação sobre a operação de dimensionamento em andamento.

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [alta disponibilidade](#)
- Saiba mais sobre [backup e recuperação](#)

Reiniciar o Banco de Dados do Azure para PostgreSQL - Servidor flexível

09/08/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo apresenta o procedimento passo a passo para executar a reinicialização do servidor flexível. Essa operação é útil para aplicar alterações de parâmetro estático que exigem a reinicialização do servidor de banco de dados. O procedimento é o mesmo para servidores configurados com alta disponibilidade com redundância de zona.

IMPORTANT

Quando configurados com alta disponibilidade, os servidores primários e em espera são reiniciados ao mesmo tempo.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Você precisa ter um servidor flexível.

Reinic peace seu servidor flexível

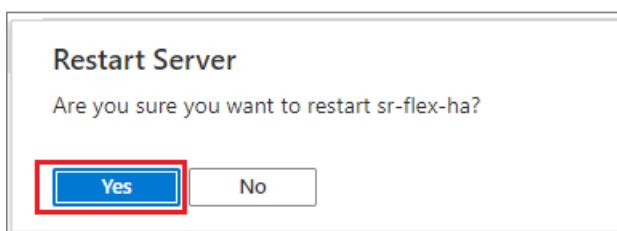
Siga estas etapas para reiniciar o servidor flexível.

1. No [portal do Azure](#), escolha o servidor flexível que você quer reiniciar.
2. Clique em **Visão geral** no painel esquerdo e clique em **Reiniciar**.

The screenshot shows the Azure portal interface for a flexible server. At the top, there's a search bar and several action buttons: Delete, Reset the password, Restore, Restart (which is highlighted with a red box), Stop, Start, and Feedback. Below the header, there's a sidebar with links: Overview (which is selected and highlighted with a red box), Activity log, Access control (IAM), and Tags. The main content area is titled 'Essentials' and displays resource group information ('my-resource'), status ('Available'), and location ('Location'). To the right, specific server details are listed: Server name (flexible-server.postgres.database.azure.com), Server admin login name (sr), and Configuration.

3. Uma mensagem pop-up de confirmação será exibida.

4. Clique em **Sim** se quiser continuar.



5. Será mostrada uma notificação de que a operação de reinicialização foi iniciada.

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [alta disponibilidade com redundância de zona](#)

Reiniciar um Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão preliminar)

09/08/2021 • 2 minutes to read

IMPORTANT

O Banco de Dados do Azure para PostgreSQL – Servidor Flexível atualmente está em versão prévia pública.

Este artigo mostra como reiniciar, iniciar e interromper o servidor flexível usando a CLI do Azure.

Pré-requisitos

- Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.
- Instale ou atualize CLI do Azure para a versão mais recente. Consulte [Instalar a CLI do Azure](#).
- Faça logon na conta do Azure usando o comando `az login`. Observe a propriedade `id`, que se refere à **ID da Assinatura** para sua conta do Azure.

```
az login
```

- Se você tiver várias assinaturas, escolha a apropriada na qual você deseja criar o servidor usando o comando `az account set .``

```
az account set --subscription <subscription id>
```

- Crie um Servidor Flexível do PostgreSQL se ainda não tiver criado um usando o comando `az postgres flexible-server create`.

```
az postgres flexible-server create --resource-group myresourcegroup --name myservername
```

Reiniciar um servidor

Para reiniciar um servidor, execute o comando `az postgres flexible-server restart`. Se você estiver usando o [contexto local](#), não precisa fornecer argumentos.

Uso:

```
az postgres flexible-server restart [--name]
                                    [--resource-group]
                                    [--subscription]
```

Exemplo sem contexto local:

```
az postgres flexible-server restart --resource-group --name myservername
```

Exemplo com contexto local:

```
az postgres flexible-server restart
```

IMPORTANT

Depois que o servidor foi reiniciado com sucesso, todas as operações de gerenciamento agora estarão disponíveis para o servidor flexível.

Próximas etapas

- Saiba mais sobre [como parar e iniciar o Servidor flexível do Banco de Dados do Azure para PostgreSQL](#)

Restauração pontual de um servidor flexível

09/08/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo fornece um procedimento passo a passo para realizar recuperações pontuais no servidor flexível usando backups. Você pode restaurar para um ponto de restauração mais recente ou um ponto de restauração personalizado dentro do período de retenção.

Pré-requisitos

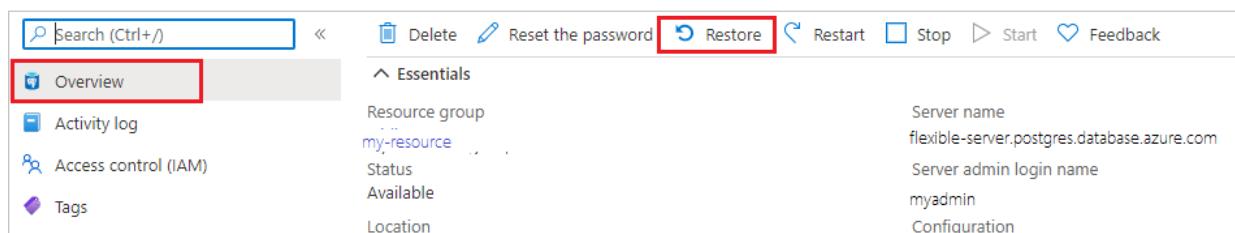
Para concluir este guia de instruções, você precisa:

- Você deve ter um Banco de Dados do Azure para PostgreSQL – Servidor Flexível. O mesmo procedimento também é aplicável para o servidor flexível configurado com redundância de zona.

Restaurar para o ponto de restauração mais recente

Siga estas etapas para restaurar o servidor flexível usando um backup existente.

1. No [portal do Azure](#), escolha o servidor flexível do qual você deseja restaurar o backup.
2. Clique em **Visão geral** no painel esquerdo e clique em **Restaurar**



3. A página Restaurar será mostrada com uma opção para escolher entre o Ponto de restauração mais recente e o Ponto de restauração personalizado.
4. Selecione **Ponto de restauração mais recente** e forneça um novo nome do servidor no campo **Restaurar para o novo servidor**. Opcionalmente, você pode escolher a zona de disponibilidade para a qual restaurar.

Source details

Select a backup source and detail. Additional settings will be defaulted where possible based on the backup selected.

Source server ⓘ



Earliest restore point ⓘ

2021-04-18T18:36:00.000Z UTC

Point-in-time-restore (PITR) ⓘ

Latest restore point (Now)

Select a custom restore point

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Name * ⓘ

Location ⓘ

East US

PostgreSQL version ⓘ

12

Availability zone ⓘ

1

Compute + storage ⓘ

General Purpose, D2s_v3

2 vCores, 8 GiB RAM, 128 GiB storage

5. Clique em OK.

6. Aparecerá uma notificação mostrando que a operação de restauração foi iniciada.

Restaurar para um ponto de restauração personalizado

Siga estas etapas para restaurar o servidor flexível usando um backup existente.

1. No [portal do Azure](#), escolha o servidor flexível do qual você deseja restaurar o backup.

2. Na página de visão geral, clique em **Restaurar**.

The screenshot shows the Azure portal interface for a flexible server named 'flexible-server.postgres.database.azure.com'. The 'Overview' tab is selected, indicated by a red box around its icon. The 'Restore' button, also indicated by a red box, is located in the top navigation bar. To the right, there's a summary section titled 'Essentials' with fields for Resource group, Status, Location, Server name, Server admin login name, and Configuration.

3. A página Restaurar será mostrada com uma opção para escolher entre o Ponto de restauração mais recente e o Ponto de restauração personalizado.

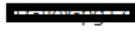
4. Escolha **Ponto de restauração personalizado**.

5. Selecione data e hora e forneça um novo nome do servidor no campo **Restaurar para o novo servidor**. Forneça um novo nome de servidor e, opcionalmente, você pode escolher a **zona de disponibilidade** para a qual restaurar.

Source details

Select a backup source and detail. Additional settings will be defaulted where possible based on the backup selected.

Source server ⓘ



Earliest restore point ⓘ

2021-04-18T18:36:00.000Z UTC

Point-in-time-restore (PITR) ⓘ

Latest restore point (Now)

Select a custom restore point

Custom restore point (UTC) ⓘ

04/20/2021

2:55:56 AM

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Name * ⓘ

mynewserver



Location ⓘ

East US



PostgreSQL version ⓘ

12



Availability zone ⓘ

1



Compute + storage ⓘ

General Purpose, D2s_v3

2 vCores, 8 GiB RAM, 128 GiB storage

6. Clique em OK.

7. Aparecerá uma notificação mostrando que a operação de restauração foi iniciada.

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [alta disponibilidade com redundância de zona](#)
- Saiba mais sobre [backup e recuperação](#)

Recuperação pontual de um Banco de Dados do Azure para PostgreSQL – Servidor Flexível com a CLI do Azure

09/08/2021 • 2 minutes to read

IMPORTANT

O Banco de Dados do Azure para PostgreSQL – Servidor Flexível atualmente está em versão preliminar pública.

Este artigo fornece um procedimento passo a passo para realizar recuperações pontuais no servidor flexível usando backups.

Pré-requisitos

- Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.
- Instale ou atualize CLI do Azure para a versão mais recente. Consulte [Instalar a CLI do Azure](#).
- Faça logon na conta do Azure usando o comando `az login`. Observe a propriedade `id`, que se refere à **ID da Assinatura** para sua conta do Azure.

```
az login
```

- Se você tiver várias assinaturas, escolha a apropriada na qual você deseja criar o servidor usando o comando `az account set .``

```
az account set --subscription <subscription id>
```

- Crie um Servidor Flexível do PostgreSQL se ainda não tiver criado um usando o comando `az postgres flexible-server create .`

```
az postgres flexible-server create --resource-group myresourcegroup --name myservername
```

Restaurar um servidor do backup para um novo servidor

Você pode executar o comando a seguir para restaurar um servidor para um backup mais antigo existente.

Usage

```
az postgres flexible-server restore --restore-time  
      --source-server  
      [--ids]  
      [--location]  
      [--name]  
      [--no-wait]  
      [--resource-group]  
      [--subscription]
```

Exemplo: restaurar um servidor deste instantâneo de backup `2021-03-03T13:10:00Z`.

```
az postgres server restore \
--name mydemoserver-restored \
--resource-group myresourcegroup \
--restore-point-in-time "2021-05-05T13:10:00Z" \
--source-server mydemoserver
```

O tempo necessário para restaurar dependerá do tamanho dos dados armazenados no servidor.

Executar tarefas de pós-restauração

Após concluir a restauração, você deve realizar as seguintes tarefas para colocar os usuários e os aplicativos novamente em execução:

- Se o novo servidor é usado para substituir o servidor original, redirecione clientes e aplicativos de cliente para o novo servidor
- Verifique se as regras de VNet adequadas estão em vigor para que os usuários se conectem. Essas regras não são copiadas do servidor original.
- Verifique se as permissões e os logons adequados no nível do banco de dados estão em vigor
- Configurar alertas conforme apropriado para o servidor de restauração recente

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [backup e recuperação](#)

Parar/iniciar um Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão preliminar) usando o portal do Azure

09/08/2021 • 2 minutes to read

IMPORTANT

Banco de Dados do Azure para PostgreSQL - O Servidor Flexível está atualmente na versão prévia.

Este artigo fornece instruções passo a passo para parar e iniciar um servidor flexível.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Você deve ter um Servidor Flexível do Banco de Dados do Azure para PostgreSQL.

Parar um servidor em execução

1. Na [portal do Azure](#), escolha o servidor flexível que você deseja parar.
2. Na página **Visão geral**, clique no botão **Parar** na barra de ferramentas.

NOTE

Depois que o servidor for interrompido, as outras operações de gerenciamento não estarão disponíveis para o servidor flexível.

Observe que os servidores interrompidos serão automaticamente ativados novamente após sete dias. Todas as atualizações de manutenção pendentes serão aplicadas quando o servidor for iniciado na próxima vez.

Iniciar um servidor interrompido

1. Na [portal do Azure](#), escolha o servidor flexível que você deseja iniciar.
2. Na página **Visão geral**, clique no botão **Iniciar** na barra de ferramentas.

NOTE

Depois que o servidor for iniciado, as outras operações de gerenciamento estarão disponíveis para o servidor flexível.

Próximas etapas

- Saiba mais sobre [opções de processamento e armazenamento no Servidor Flexível do Banco de Dados do Azure para PostgreSQL](#).

Parar/iniciar o Banco de Dados do Azure para PostgreSQL – Servidor Flexível (versão preliminar) usando a CLI do Azure

09/08/2021 • 2 minutes to read

IMPORTANT

O Banco de Dados do Azure para PostgreSQL – Servidor Flexível atualmente está em versão prévia pública.

Este artigo mostra como reiniciar, iniciar e interromper o servidor flexível usando a CLI do Azure.

Pré-requisitos

- Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.
- Instale ou atualize CLI do Azure para a versão mais recente. Consulte [Instalar a CLI do Azure](#).
- Faça logon na conta do Azure usando o comando `az login`. Observe a propriedade `id`, que se refere à [ID da Assinatura](#) para sua conta do Azure.

```
az login
```

- Se você tiver várias assinaturas, escolha apropriada na qual você deseja criar o servidor usando o comando `az account set`.

```
az account set --subscription <subscription id>
```

- Crie um Servidor Flexível do PostgreSQL se ainda não tiver criado um usando o comando `az postgres flexible-server create`.

```
az postgres flexible-server create --resource-group myresourcegroup --name myservername
```

Interromper um servidor em execução

Para interromper um servidor, execute o comando `az postgres flexible-server stop`. Se você estiver usando o [contexto local](#), não precisa fornecer argumentos.

Uso:

```
az postgres flexible-server stop [--name]
                                [--resource-group]
                                [--subscription]
```

Exemplo sem contexto local:

```
az postgres flexible-server stop --resource-group --name myservername
```

Exemplo com contexto local:

```
az postgres flexible-server stop
```

Iniciar um servidor interrompido

Para iniciar um servidor, execute o comando `az postgres flexible-server start`. Se você estiver usando o [contexto local](#), não precisa fornecer argumentos.

Uso:

```
az postgres flexible-server start [--name]
                                  [--resource-group]
                                  [--subscription]
```

Exemplo sem contexto local:

```
az postgres flexible-server start --resource-group --name myservername
```

Exemplo com contexto local:

```
az postgres flexible-server start
```

IMPORTANT

Depois que o servidor foi reiniciado com sucesso, todas as operações de gerenciamento agora estarão disponíveis para o servidor flexível.

Próximas etapas

- Saiba mais sobre a [reiniciar o Servidor flexível do Banco de Dados do Azure para PostgreSQL](#)

Gerencie a alta disponibilidade com redundância de zona no Servidor Flexível

09/08/2021 • 5 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo descreve como você pode habilitar ou desabilitar a configuração de alta disponibilidade com redundância de zona no seu servidor flexível.

O recurso de alta disponibilidade provisiona a réplica primária e em espera fisicamente separada em zonas diferentes. Para obter mais detalhes, confira a [documentação de conceitos de alta disponibilidade](#). Você pode optar por habilitar a alta disponibilidade no momento da criação do servidor flexível ou após a criação. Esta página fornece diretrizes de como você pode habilitar ou desabilitar a alta disponibilidade. Essa operação não altera as outras configurações, incluindo configuração de VNet, configurações de firewall e retenção de backup. De modo similar, habilitar e desabilitar a alta disponibilidade é uma operação online e não afeta a conectividade e as operações do aplicativo.

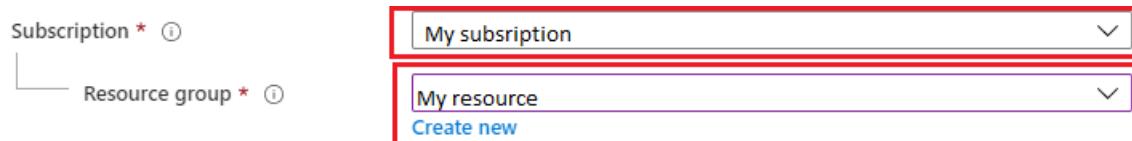
Pré-requisitos

A alta disponibilidade redundante de zona está disponível apenas em regiões onde há suporte para várias zonas.

Habilitar alta disponibilidade durante a criação do servidor

Esta seção apresenta detalhes especificamente para campos relacionados à HA. Você pode seguir estas etapas para implantar a alta disponibilidade ao criar seu servidor flexível.

1. No [portal do Azure](#), escolha Servidor Flexível e clique em criar. Para obter detalhes sobre como preencher detalhes como **Assinatura**, **Grupo de recursos**, **nome do servidor**, **região** e outros campos, confira a documentação de instruções para a criação do servidor.



Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name *	<input type="text" value="my server name"/>
Region *	<input type="text" value="East US"/>

2. Escolha sua **zona de disponibilidade**. Isso será útil se você quiser colocar seu aplicativo na mesma zona de disponibilidade que o banco de dados para reduzir a latência. Escolha **Sem Preferência** se quiser que o servidor flexível implante em qualquer zona de disponibilidade.

Availability zone

No preference

No preference

1

2

3

3. Clique na caixa de seleção de **Alta disponibilidade com redundância de zona** na opção Disponibilidade.

Availability option Zone redundant high availability

4. Se você quiser alterar a computação e o armazenamento padrão, clique em **Configurar servidor**.

Compute + storage ⓘ

General Purpose, D4s_v3

4 vCores, 16 GiB RAM, 512 GiB storage

[Configure server](#)

5. Se a opção de alta disponibilidade estiver marcada, a camada com capacidade de intermitência não estará disponível para escolher. Você pode escolher as camadas de computação de **uso geral** ou **otimizado para memória**. Então, você pode selecionar o **tamanho da computação** para sua escolha no menu suspenso.

Compute tier

- Burstable (1-2 vCores) - Best for workloads that don't need the full CPU continuously
- General Purpose (2-64 vCores) - Balanced configuration for most common workloads
- Memory Optimized (2-64 vCores) - Best for workloads that require a high memory to CPU ratio

Selection is disabled because zone redundant high availability is not supported with burstable compute tier.

Compute size

Standard_D4s_v3 (4 vCores, 16 GB memory)

6. Selecione o **tamanho do armazenamento** em GiB usando a barra deslizante e selecione o **período de retenção de backup** entre 7 e 35 dias.

Storage size (in GiB)



Storage cannot be scaled down

UP TO 5000 AVAILABLE IOPS

High availability

Enable high availability to have a standby replica for your server with automatic failover.

Availability option Enabled (Zone redundant)

Backups

Configure automatic server backups that can be used to restore your server to a point-in-time.

Retention period



7. Clique em **Salvar**.

Habilitar a criação de servidor de postagem de alta disponibilidade

Siga estas etapas para habilitar a alta disponibilidade para seu servidor flexível existente.

1. No [portal do Azure](#), selecione o servidor flexível PostgreSQL existente.
2. Na página do servidor flexível, clique em **Alta Disponibilidade** no painel esquerdo para abrir a página alta disponibilidade.

The screenshot shows the Azure portal interface for a PostgreSQL flexible server named 'flex-server'. The left sidebar has a 'High availability' section with several options: Compute + storage, Networking, Connection strings, Server parameters, Maintenance Window, and High availability. The 'High availability' option is highlighted with a red box. Other sections like Overview, Activity log, Access control (IAM), and Tags are also visible in the sidebar.

3. Clique na caixa de seleção **alta disponibilidade com redundância de zona** para habilitar a opção e clique em **Salvar** para salvar a alteração.

The screenshot shows the Azure portal interface for managing a PostgreSQL flexible server named 'flex-server'. The left sidebar has a 'Settings' section with several options: Compute + storage, Networking, Connection strings, Server parameters, Maintenance Window, High availability (which is selected and highlighted with a grey background), and Locks. The main content area is titled 'High availability' and contains the instruction: 'Enable high availability to have a standby replica for your server with automatic failover.' Below this, there's a section for 'Availability option' where 'Zone redundant high availability' is checked and highlighted with a red box. At the bottom, it shows 'Availability zone Zone - 2'. At the very top, there are 'Search (Ctrl+)', 'Save' (highlighted with a red box), and 'Discard' buttons.

4. Uma caixa de diálogo de confirmação mostrará que, habilitando a alta disponibilidade, seu custo aumentará devido à implantação adicional do servidor e do armazenamento.
5. Clique no botão **Habilitar HA** para habilitar a alta disponibilidade.
6. Uma notificação será exibida mostrando que a implantação da alta disponibilidade está em andamento.

Desabilitar alta disponibilidade

Siga estas etapas para desabilitar a alta disponibilidade para seu servidor flexível que já está configurado com redundância de zona.

1. No [portal do Azure](#), selecione o atual Servidor Flexível do Banco de Dados do Azure para PostgreSQL.
2. Na página servidor flexível, clique em **Alta Disponibilidade** no painel frontal para abrir a página alta disponibilidade.



flex-server

Azure Database for PostgreSQL flexible server

Search (Ctrl+ /) <<

- Overview
- Activity log
- Access control (IAM)
- Tags

Settings

- Compute + storage
- Networking
- Connection strings
- Server parameters
- Maintenance Window
- High availability**
- Locks

Monitoring

- Alerts
- Metrics
- Diagnostic settings
- Logs

3. Clique na caixa de seleção **alta disponibilidade com redundância de zona** para desabilitar a opção.
Em seguida, clique em **Salvar** para salvar a alteração.

flex-server | High availability

Azure Database for PostgreSQL flexible server

Search (Ctrl+ /) << **Save** Discard

- Overview
- Activity log
- Access control (IAM)
- Tags

High availability

Enable high availability to have a standby replica for your server with automatic failover.

Availability option Zone redundant high availability

Availability zone Zone - 1

Settings

- Compute + storage
- Networking
- Connection strings
- Server parameters
- Maintenance Window
- High availability**
- Locks

4. Será mostrada uma caixa de diálogo de confirmação na qual você pode confirmar a desabilitação da alta disponibilidade.

5. Clique no botão **Desabilitar HA** para desabilitar a alta disponibilidade.
6. Uma notificação será exibida descomissionando a implantação de alta disponibilidade em andamento.

failover forçado

Siga estas etapas para forçar o failover do seu primário para o servidor flexível em espera. Isso faz o primário se tornar imediatamente inoperante, e dispara um failover para o servidor em espera. Isso é útil para casos como testar o tempo de failover de interrupção não planejada para sua carga de trabalho.

1. No [portal do Azure](#), selecione o servidor flexível existente que tenha o recurso de alta disponibilidade já habilitado.
2. Na página servidor flexível, clique em Alta Disponibilidade no painel frontal para abrir a página alta disponibilidade.
3. Verifique a zona de disponibilidade Primária e a zona de disponibilidade Em espera
4. Clique em Failover Forçado para iniciar o procedimento de failover manual. Um pop-up informará sobre o possível tempo de inatividade até que o failover seja concluído. Leia a mensagem e clique em OK.
5. Uma notificação aparecerá mencionando que o failover está em andamento.
6. Depois que o failover para o servidor em espera estiver concluído, surgirá uma notificação.
7. Verifique a nova zona de disponibilidade Primária e a zona de disponibilidade Em espera.

The screenshot shows the Azure portal interface for managing a flexible server's high availability settings. The 'High availability' blade is open. At the top, there are buttons for 'Save', 'Discard', 'Forced Failover' (which is highlighted with a red box), 'Planned Failover', and 'Feedback'. Below these, a note states: 'Zone redundant high availability deploys a standby replica in a different zone with automatic failover capability.' A 'Learn more' link is provided. Under the 'High availability' section, there are two dropdown menus: 'Primary availability zone' set to 1 and 'Standby availability zone' set to 2. Both dropdowns are also highlighted with red boxes. On the left sidebar, under the 'Settings' section, the 'High availability' option is selected and highlighted with a red box.

IMPORTANT

- Não execute failovers sucessivos e imediatos. Aguarde pelo menos de 15 a 20 minutos entre os failovers, o que também permitirá que o novo servidor em espera seja totalmente estabelecido.
- O tempo de operação geral de ponta a ponta, conforme relatado no portal, pode ser maior do que o tempo de inatividade real apresentado pelo aplicativo. Considere o tempo de inatividade pela perspectiva do aplicativo.

Failover planejado

Siga estas etapas para executar um failover planejado do seu servidor principal para o servidor flexível em espera. Isso primeiro prepara o servidor em espera e executa o failover. Isso gera o menor tempo de inatividade, pois executa um failover normalmente para o servidor em espera para situações como após um evento de failover em que você queira trazer o primário de volta para a zona de disponibilidade preferencial.

1. No [portal do Azure](#), selecione o servidor flexível existente que tenha o recurso de alta disponibilidade já habilitado.

2. Na página servidor flexível, clique em Alta Disponibilidade no painel frontal para abrir a página alta disponibilidade.
3. Verifique a zona de disponibilidade Primária e a zona de disponibilidade Em espera
4. Clique em Failover Planejado para iniciar o procedimento de failover manual. Um pop-up informará o processo. Leia a mensagem e clique em OK.
5. Uma notificação aparecerá mencionando que o failover está em andamento.
6. Depois que o failover para o servidor em espera estiver concluído, surgirá uma notificação.
7. Verifique a nova zona de disponibilidade Primária e a zona de disponibilidade Em espera.

The screenshot shows the Azure portal interface for managing a flexible server's high availability settings. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Settings, Compute + storage, Networking, Connection strings, Server parameters, Maintenance, and High availability. The 'High availability' option is highlighted with a red box. The main content area displays the current configuration: 'High availability' is selected (indicated by a checked checkbox), and 'Zone redundant HA' is also checked. The 'High availability status' is listed as 'Healthy'. Below that, it shows 'Primary availability zone' set to 1 and 'Standby availability zone' set to 2, both of which are also highlighted with a red box. At the top of the page, there are buttons for Save, Discard, Forced Failover, Planned Failover (which is highlighted with a red box), and Feedback.

IMPORTANT

- Não execute failovers sucessivos e imediatos. Aguarde pelo menos de 15 a 20 minutos entre os failovers, o que também permitirá que o novo servidor em espera seja totalmente estabelecido.
- É recomendável executar o failover planejado durante um período de baixa atividade.
- O tempo de operação geral de ponta a ponta pode ser maior do que o tempo de inatividade real apresentado pelo aplicativo. Considere o tempo de inatividade pela perspectiva do aplicativo.

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [alta disponibilidade com redundância de zona](#)

Gerenciar a alta disponibilidade com redundância de zona no servidor flexível do Banco de Dados do Azure para PostgreSQL com a CLI do Azure

09/08/2021 • 2 minutes to read

NOTE

O servidor flexível do Banco de Dados do Azure para PostgreSQL está na fase de pré-visualização pública.

O artigo descreve como você pode habilitar ou desabilitar a configuração de alta disponibilidade redundante da zona no momento da criação do servidor em seu servidor flexível. Você também pode desabilitar a alta disponibilidade com redundância de zona após a criação do servidor. Não há suporte para a habilitação da alta disponibilidade com redundância de zona após a criação do servidor.

O recurso de alta disponibilidade provisiona a réplica primária e em espera fisicamente separada em zonas diferentes. Para obter mais detalhes, confira a [documentação de conceitos de alta disponibilidade](#). Habilitar ou desabilitar a alta disponibilidade não altera suas outras configurações, incluindo configuração de VNET, configurações de firewall e retenção de backup. Desabilitar a alta disponibilidade não afeta a conectividade e as operações do aplicativo.

IMPORTANT

Para ver a lista de regiões que suportam alta disponibilidade com redundância de zona, consulte as regiões com suporte [aqui](#).

Pré-requisitos

- Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.
- Instale ou atualize CLI do Azure para a versão mais recente. Consulte [Instalar a CLI do Azure](#).
- Faça logon na conta do Azure usando o comando `az login`. Observe a propriedade `id`, que se refere à [ID da Assinatura](#) para sua conta do Azure.

```
az login
```

- Se você tiver várias assinaturas, escolha apropriada na qual você deseja criar o servidor usando o comando `az account set`.

```
az account set --subscription <subscription id>
```

Habilitar alta disponibilidade durante a criação do servidor

Você só pode criar um servidor usando os tipos de preço uso geral ou otimizado para memória com alta disponibilidade. Você pode habilitar a alta disponibilidade para um servidor somente durante o tempo de criação.

Uso:

```
az postgres flexible-server create [--high-availability {Disabled, Enabled}]  
    [--resource-group]  
    [--name]
```

Exemplo:

```
az postgres flexible-server create --name myservername --sku-name Standard-D2ds_v4 --resource-group  
myresourcegroup --high-availability Enabled
```

Desabilitar alta disponibilidade

Você pode desabilitar a alta disponibilidade usando o comando [az postgres flexible-server update](#). Observe que só haverá suporte para desabilitar a alta disponibilidade se o servidor estiver configurado com ela.

```
az postgres flexible-server update [--high-availability {Disabled, Enabled}]  
    [--resource-group]  
    [--name]
```

Exemplo:

```
az postgres flexible-server update --resource-group myresourcegroup --name myservername --high-availability  
Disabled
```

Próximas etapas

- Saiba mais sobre [continuidade dos negócios](#)
- Saiba mais sobre [alta disponibilidade com redundância de zona](#)

Usar o portal do Azure para configurar alertas nas métricas do Banco de Dados do Azure para PostgreSQL – Servidor Flexível

09/08/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Este artigo mostra como configurar alertas do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure. Você pode receber um alerta com base em métricas de monitoramento para seus serviços do Azure.

O alerta é disparado quando o valor de uma métrica especificada ultrapassa um limite atribuído por você. Ele dispara tanto quando a condição é atendida pela primeira vez e quanto posteriormente, quando essa condição não está mais sendo atendida.

Você pode configurar um alerta para fazer as seguintes ações quando ele disparar:

- Enviar notificações por email para o administrador e coadministradores de serviços.
- Enviar um email para outros emails que você especificar.
- Chamar um webhook.

Você pode configurar e obter informações sobre as regras de alerta usando:

- [Azure portal](#)
- [CLI do Azure](#)
- [API REST do Azure Monitor](#)

Criar uma regra de alerta em uma métrica no Portal do Azure

1. No [Portal do Azure](#), selecione o servidor do Banco de Dados do Azure para PostgreSQL que você deseja monitorar.
2. Na seção **Monitoramento** da barra lateral, selecione **Alertas** como mostrado abaixo:

3. Selecione **Adicionar alerta de métrica** (ícone +).

4. A página **Criar regra** é aberta, conforme mostrado abaixo. Preencha as informações obrigatórias:

5. Dentro da seção **Condição**, selecione **Adicionar condição**.

6. Selecione uma métrica da lista de sinais sobre a qual deseja ser alertado. Neste exemplo, selecione "Porcentagem de armazenamento".

Create rule
Rules management

RESOURCE
mydemouser > mysubscription > myresourcegroup

HIERARCHY

CONDITION
No condition defined, click on 'Add condition' to select a signal and define its logic

ACTION GROUPS
Notify your team via email and text messages or automate actions using webhooks, runbooks, functions, logic a integrating with external ITSM solutions. Learn more [here](#)

ACTION GROUP NAME ACTION GROUP TYPE

No action group selected
[Select existing](#) [Create New](#)

ALERT DETAILS

- * Alert rule name [i](#)
Specify alert rule name. Sample: 'Percentage CPU greater than 70'
- * Description
Specify alert description here...

[Create alert rule](#)

Configure signal logic

Choose a signal below and configure the logic on the next screen to define the alert condition.

All signals (46)

SIGNAL NAME	SIGNAL TYPE	MONITOR SERVICE
CPU percent	Metric	Platform
Memory percent	Metric	Platform
IO percent	Metric	Platform
Storage percent	Metric	Platform
Storage used	Metric	Platform
Storage limit	Metric	Platform
Server Log storage percent	Metric	Platform
Server Log storage used	Metric	Platform
Server Log storage limit	Metric	Platform
Active Connections	Metric	Platform
Failed Connections	Metric	Platform
Backup Storage used	Metric	Platform
Network Out	Metric	Platform
Network In	Metric	Platform
Replica Lag	Metric	Platform

[Done](#)

7. Configure a lógica de alerta, incluindo a **Condição** (por exemplo, "Maior que"), o **Limite** (por exemplo, 85%), a **Agregação de Tempo**, o **Período** durante o qual a regra de métrica deverá ser atendida antes de o alerta disparar (por exemplo, "Os últimos 30 minutos") e **Frequência**.

Selecione **Concluído** ao concluir.

Create rule
Rules management

RESOURCE
mydemouser > mysubscription > myresourcegroup

HIERARCHY

CONDITION
No condition defined, click on 'Add condition' to select a signal and define its logic

ACTION GROUPS
Notify your team via email and text messages or automate actions using webhooks, runbooks, functions, logic a integrating with external ITSM solutions. Learn more [here](#)

ACTION GROUP NAME ACTION GROUP TYPE

No action group selected
[Select existing](#) [Create New](#)

ALERT DETAILS

- * Alert rule name [i](#)
Specify alert rule name. Sample: 'Percentage CPU greater than 70'
- * Description
Specify alert description here...

[Create alert rule](#)

Configure signal logic

Storage percent(Platform)

Show history
Over the last 6 hours

10%
8%
6%
4%
2%
0%
8 AM 9 AM 10 AM 11 AM 12 PM 1 PM

Storage percent (Avg)
mydemouser
10.43 %

Alert logic

Condition [i](#) * Time Aggregation [i](#) * Threshold [i](#)
Greater than Average 85 %

Condition preview
Whenever the storage percent is greater than 85 percent

Evaluated based on

Period (grain) [i](#) Frequency [i](#)
Over the last 30 minutes Every 1 Minute

[Done](#)

8. Dentro da seção **Grupos de Ações**, selecione **Criar Novo** para criar um novo grupo para receber notificações sobre o alerta.
9. Preencha o formulário "Adicionar grupo de ações" com um nome, o nome curto, a assinatura e o grupo de recursos.
10. Configure o tipo de ação **Email/SMS/Push/Voz**.
 - a. Escolha "Enviar email para a Função do Azure Resource Manager" para selecionar os Proprietários da assinatura, Colaboradores e Leitores para receber notificações.

b. Opcionalmente, forneça um URI válido no campo **Webhook** se você quiser chamá-lo quando o alerta for disparado.

c. Selecione **OK** ao concluir.

The screenshot shows two overlapping windows. The top window is titled 'Add action group' and contains fields for 'Action group name' (newactiongroup), 'Short name' (group1), 'Subscription' (mysubscription), and 'Resource group' (myresourcegroup). It also has an 'Actions' section with a table showing one row: 'newactiongroup' (ACTION NAME), 'Email/SMS/Push/V...' (ACTION TYPE), and 'Edit details' (DETAILS). A note says 'Please configure the action by clicking the link.' Below this is a 'Privacy Statement' and a 'Pricing' link. The bottom window is titled 'Email/SMS/Push/Voice' and shows configuration for an email action. It includes fields for 'Name' (email), 'Email' (email@contoso.com), 'SMS' (Country code 1, Phone number 1234567890), and 'Voice' (Country code 1, Phone number 1234567890). There is also a note about carrier charges and a note that push notifications are only supported for Service Health alerts. Both windows have an 'OK' button at the bottom right.

11. Especifique um Nome da regra de alerta, uma Descrição e uma Gravidade.

The screenshot shows the 'Create rule' dialog in the Azure portal. It has a sidebar with various icons. The main area is titled 'Create rule' and 'Rules management'. It has sections for 'ACTION GROUPS' (with a note about notifying teams via email and webhooks) and 'ALERT DETAILS'. The 'ALERT DETAILS' section is highlighted with a red box and contains fields for 'Alert rule name' (Storage percentage greater than 85), 'Description' (Storage percentage greater than 85), 'Severity' (Sev 3), and 'Enable rule upon creation' (Yes). At the bottom is a note about rule activation and a 'Create alert rule' button.

12. Selecione **Criar regra de alerta** para criar o alerta.

Em alguns minutos, o alerta estará ativo e disparará conforme descrito anteriormente.

Gerenciar seus alertas

Depois de criar um alerta, você poderá selecioná-lo e executar as seguintes ações:

- Exibir um grafo mostrando o limite de métrica e os valores reais do dia anterior relevante para este alerta.
- **Editar** ou **Excluir** a regra de alerta.
- **Desabilitar** ou **Habilitar** o alerta, se desejar interromper temporariamente ou retomar o recebimento de notificações.

Próximas etapas

- Saiba mais sobre como [configurar webhooks em alertas](#).
- Tenha uma [visão geral da coleção de métricas](#) para verificar se o serviço está disponível e responsivo.

Configurar e acessar logs no Banco de Dados do Azure para PostgreSQL – Servidor Flexível

09/08/2021 • 2 minutes to read

IMPORTANT

O Servidor Flexível do Banco de Dados do Azure para PostgreSQL está em versão prévia

Os logs do PostgreSQL estão disponíveis em todos os nós de um servidor flexível. Você pode enviar os logs para um servidor de armazenamento ou para um serviço de análise. Esses logs podem ser usados para identificar, solucionar problemas e reparar erros de configuração e desempenho abaixo do ideal.

Definir as configurações de diagnóstico

Para habilitar as configurações de diagnóstico para o servidor Postgres, use o portal do Azure, a CLI, a API REST e o PowerShell. A categoria de log a ser selecionada é **PostgreSQLLogs**.

Para habilitar os logs de recursos usando o portal do Azure:

1. No portal, vá até *Configurações de diagnóstico* no menu de navegação do servidor do Postgres.
2. Selecione *Adicionar configuração de diagnóstico*.

The screenshot shows the Azure portal interface for managing diagnostic settings for a PostgreSQL server named 'example-group-c'. The left sidebar has a red box around the 'Diagnostic settings' link under the 'Monitoring' section. The main content area shows a table with one row, 'No diagnostic settings defined'. A red box surrounds the '+ Add diagnostic setting' button. Below the table, there's a note: 'Click 'Add Diagnostic setting' above to configure the collection of the following data:' followed by a bulleted list: 'PostgreSQLLogs' and 'AllMetrics'.

3. Defina um nome para essa configuração.
4. Selecione o ponto de extremidade preferido (conta de armazenamento, Hub de eventos, análise de logs).
5. Selecione o tipo de log **PostgreSQLLogs**.

Diagnostics setting

Save Discard Delete Provide feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name *

example-group-c-logs

1

Category details

log

PostgreSQLLogs

2

metric

AllMetrics

Destination details

- Send to Log Analytics
- Archive to a storage account
- Stream to an event hub

3

6. Salve sua configuração.

Para habilitar os logs de recursos usando o PowerShell, a CLI ou a API REST, confira o artigo [configurações de diagnóstico](#).

Acessar logs de recursos

A forma de acessar os logs depende do ponto de extremidade escolhido. Para o Armazenamento do Microsoft Azure, consulte o artigo sobre a [conta de armazenamento de logs](#). Para os hubs de eventos, consulte o artigo sobre [fluxos de logs do Azure](#).

Para logs de Azure Monitor, os logs são enviados para o espaço de trabalho selecionado. Os logs do Postgres usam o modo de coleta **AzureDiagnostics** para que possam ser consultados da tabela do AzureDiagnostics. Os campos na tabela são descritos abaixo. Saiba mais sobre como consultar e alertar na visão geral [Consulta de logs do Azure Monitor](#).

Para começar, execute as consultas a seguir. Você pode configurar alertas com base em consultas.

Pesquisar todos os logs do Postgres para um servidor específico no último dia

```
AzureDiagnostics
| where LogicalServerName_s == "myservername"
| where Category == "PostgreSQLLogs"
| where TimeGenerated > ago(1d)
```

Pesquisar todas as tentativas de conexão não localhost

```
AzureDiagnostics
| where Message contains "connection received" and Message !contains "host=127.0.0.1"
| where Category == "PostgreSQLLogs" and TimeGenerated > ago(6h)
```

A consulta acima mostrará os resultados nas últimas 6 horas de qualquer log de servidor Postgres nesse workspace.

Próximas etapas

- [Introdução às consultas de análise de logs](#)
- Saiba mais sobre os [Hubs de eventos do Azure](#)

O que é a Hiperescala (Citus) do Banco de Dados do Azure para PostgreSQL?

21/05/2021 • 2 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço de banco de dados relacional na nuvem da Microsoft projetado para desenvolvedores. Ele é baseado na versão da comunidade do mecanismo de banco de dados [PostgreSQL](#) de software livre.

A Hiperescala (Citus) é uma opção de implantação que escala horizontalmente as consultas em vários computadores por meio da fragmentação. Seu mecanismo de consulta faz a correspondência entre consultas SQL recebidas nesses servidores para obter respostas mais rápidas em grandes conjuntos de dados. Ele atende aplicativos que exigem maior escala e desempenho do que outras opções de implantação: geralmente, cargas de trabalho que estão se aproximando dos 100 GB de dados ou que já excederam esse limite.

A Hiperescala (Citus) fornece:

- Dimensionamento horizontal entre vários computadores usando a fragmentação
- Consultar a paralelização nesses servidores para obter respostas mais rápidas em grandes conjuntos de dados
- Excelente suporte para aplicativos multilocatário, análise operacional em tempo real e cargas de trabalho transacionais de alta taxa de transferência

Os aplicativos criados para o PostgreSQL podem executar consultas distribuídas em Hiperescala (Citus) com [bibliotecas de conexão](#) padrão e alterações mínimas.

Próximas etapas

- Comece pela [criação do seu primeiro](#) grupo de servidores da Hiperescala (Citus) do Banco de Dados do Azure para PostgreSQL.
- Consultar a [página de preços](#) para ver comparações de custo e calculadoras. A Hiperescala (Citus) também oferece descontos de Instância Reservada pré-paga; confira as páginas de [preços da Hiperescala \(Citus\)](#) em [RI](#) para obter detalhes.
- Determinar o melhor [tamanho inicial](#) para seu grupo de servidores

Versão prévia dos recursos para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 2 minutes to read

Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) oferece versão prévia para recursos não lançados. As versões prévias dos recursos são fornecidas sem um contrato de nível de serviço e não são recomendadas para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos. Para saber mais, confira os [Termos de Uso Complementares das Versões Prévias do Microsoft Azure](#)

Recursos atualmente em versão prévia

Estes são os recursos disponíveis atualmente em versão prévia:

- [Camada básica](#) . Execute um grupo de servidores usando apenas um nó coordenador e nenhum nó de trabalho. Uma forma econômica de fazer testes e desenvolvimento iniciais e lidar com cargas de trabalho de produção pequenas.
- [PostgreSQL 12 e 13](#) . Use a versão mais recente do banco de dados no seu grupo de servidores.
- [Citus 10](#) . Instalado automaticamente em grupos de servidores que executam o PostgreSQL 13.
- [Armazenamento em coluna](#) . Armazena colunas de tabelas selecionadas (ao invés de linhas) de maneira contígua no disco. Dá suporte à compactação em disco. É boa para cargas de trabalho analíticas e de data warehouse.
- [Rélicas de leitura](#) (no momento, somente da mesma região). Todas as alterações que ocorrem no grupo de servidores primário são refletidas na réplica e as consultas na réplica não causam nenhuma carga extra no original. As réplicas são uma ferramenta útil para aprimorar o desempenho de cargas de trabalho de somente leitura.
- [PgBouncer gerenciados](#) . Um pooler de conexões que permite que muitos clientes se conectem ao grupo de servidores de uma só vez e limita o número de conexões ativas. Ele atende às solicitações de conexão enquanto mantém o nó coordenador funcionando sem problemas.
- [pgAudit](#) . Fornece o log de auditoria detalhado de sessão e objeto por meio do recurso de log do PostgreSQL padrão. Ele produz logs de auditoria necessários para passar determinadas auditorias de certificação governamentais, financeiras ou ISO.

Regiões disponíveis para versão prévia dos recursos

A extensão pgAudit está disponível em todas as [regiões com suporte da Hiperescala \(Citus\)](#). As outras versões prévias dos recursos estão disponíveis apenas no leste dos EUA.

Meu grupo de servidores tem acesso à versão prévia dos recursos?

Para determinar a versão prévia dos recursos está habilitada para o grupo de servidores Hiperescala (Citus), navegue até a página [Visão Geral](#) do grupo de servidores no portal do Azure. Se você vir a propriedade **Camada: Básica (versão prévia)** ou a **Camada: Padrão (versão prévia)**, o grupo de servidores terá acesso à versão prévia dos recursos.

Como posso obter acesso

Ao criar um grupo de servidores Hiperescala (Citus), marque a caixa **Habilitar versão prévia dos recursos**.

Fale conosco

Conte-nos sobre sua experiência usando a versão prévia dos recursos enviando um email para [Pergunte ao BD do Azure para PostgreSQL](#). (Este endereço de email não é um canal de atendimento de suporte técnico. Para problemas técnicos, abra uma [solicitação de suporte](#)).

Criar um grupo de servidores de camada básica de Hiperescala (Citus) no portal do Azure

21/05/2021 • 4 minutes to read

O Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) é um serviço gerenciado usado para executar, gerenciar e dimensionar bancos de dados altamente disponíveis do PostgreSQL na nuvem. A [camada básica](#) desse serviço é uma opção de implantação conveniente para desenvolvimento e teste iniciais.

Este Guia de Início Rápido mostra como criar um grupo de servidores de camada básica de Hiperescala (Citus) usando o portal do Azure. Você vai provisionar o grupo de servidores e verificar se pode se conectar a ele para executar consultas.

IMPORTANT

A camada básica de Hiperescala (Citus) está atualmente em versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

Você pode ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Entre no Portal do Azure

Entre no [portal do Azure](#).

Siga estas etapas para criar um Banco de Dados do Azure para o servidor PostgreSQL:

1. Clique em **Criar um recurso** no canto superior esquerdo do portal do Azure.
2. Selecione **Bancos de Dados** na página **Novo** e selecione **Banco de Dados do Azure para PostgreSQL** na página **Bancos de Dados**.
3. Para a opção de implantação, clique no botão **Criar em grupo de servidores Hyperscale (Citus)**.
4. Preencha o formulário de detalhes sobre o novo servidor com as seguintes informações:
 - Grupo de recursos: clique no link **Criar novo** abaixo da caixa de texto desse campo. Insira um nome como **myresourcegroup**.
 - Nome do grupo de servidores: digite um nome exclusivo para o novo grupo de servidores que também poderá ser usado para um subdomínio do servidor.
 - Marque a caixa de seleção **Habilitar recursos de visualização**.
 - Nome de usuário do administrador: no momento deve ser o valor **citus** e não pode ser alterado.
 - Senha: deve ter pelo menos oito caracteres de comprimento e conter caracteres das três seguintes categorias: letras maiúsculas em inglês, letras minúsculas em inglês, números (0 – 9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
 - Localização: use a localização mais próxima dos usuários para fornecer a eles acesso mais rápido aos dados.

IMPORTANT

A senha de administrador do servidor que você especificar aqui é necessária para fazer logon no servidor e nos bancos de dados. Lembre-se ou registre essas informações para o uso posterior.

5. Clique em **Configurar grupo de servidores**.

- Selecione o botão de opção **Básico** para a Camada.
- Clique em **Salvar**.

6. Clique em **Próximo: Rede** > na parte inferior da tela.7. Na guia **Rede**, clique no botão de opção **Ponto de extremidade público**.

Hyperscale (Citus) server group

Microsoft

[Basics](#) [Networking](#) [Tags](#) [Review + create](#)Configure networking access and security for your server group. [Learn more](#) **Network connectivity**

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method

 No access Public endpoint

Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group

 No Yes[+ Add current client IP address \(207.153.12.62 \)](#) [+ Add 0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP
Firewall rule name	Start IP	End IP

8. Clique no link **+ Adicionar endereço IP do cliente atual**.

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method [\(i\)](#)

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group [\(i\)](#)

No

Yes

+ Add current client IP address (207.153.12.62) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP	
ClientIPAddress_2020-7-27_18-50-22	207.153.12.62	207.153.12.62	
Firewall rule name	Start IP	End IP	

NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Se você estiver tentando se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Se isso acontecer, você não poderá se conectar ao cluster do Hiperescala (Citus), a menos que o departamento de TI abra a porta 5432.

9. Clique em **Examinar + criar** e, em seguida, **Criar** para provisionar o servidor. O provisionamento demora alguns minutos.
10. A página será redirecionada para monitorar a implantação. Quando o status em tempo real mudar de **Sua implantação está em andamento** para **Sua implantação está concluída**, clique no item de menu **Saídas** no lado esquerdo da página.
11. A página de saídas conterá um nome do host do coordenador com um botão ao lado dele para copiar o valor para a área de transferência. Registre essas informações para uso posterior.

Conectar-se ao banco de dados usando psql

Ao criar o servidor do Banco de Dados do Azure para PostgreSQL, um banco de dados padrão chamado **citus** é criado. Para se conectar ao seu servidor de banco de dados, é necessário uma cadeia de conexão e a senha de administrador.

1. Obter a cadeia de conexão. Na página de grupo de servidor, clique no item de menu **Cadeias de conexão**. (Está em **Configurações**.) Localize a cadeia de caracteres marcada como **psql**. Ela terá o formato:

```
psql "host=hostname.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Copie a cadeia de caracteres. Você precisará substituir "{sua_senha}" pela senha administrativa escolhida anteriormente. O sistema não armazena a senha de texto sem formatação e, portanto, não é possível exibi-la na cadeia de conexão.

2. Abra uma janela de terminal no computador local.
3. No prompt, conecte-se ao servidor do Banco de Dados do Azure para PostgreSQL com o utilitário [psql](#). Passe a cadeia de conexão entre aspas, certificando-se de que contém a senha:

```
psql "host=..."
```

Por exemplo, o comando a seguir conecta-se ao nó coordenador do grupo de servidores **mydemoserver**:

```
psql "host=mydemoserver-c.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Próximas etapas

Neste guia de início rápido, você aprendeu como provisionar um grupo de servidores Hyperscale (Citus). Você conectou ele com o psql, criou um esquema e distribuiu dados.

- Siga um tutorial para [compilar aplicativos escalonáveis de multilocatário](#)
- Determinar o melhor [tamanho inicial](#) para seu grupo de servidores

Guia de Início Rápido – Criar um grupo de servidores Hiperescala (Citus) no portal do Azure

21/05/2021 • 7 minutes to read

O Banco de Dados do Azure para PostgreSQL é um serviço gerenciado usado para executar, gerenciar e dimensionar bancos de dados altamente disponíveis do PostgreSQL na nuvem. Este Guia de Início Rápido mostra como criar um Banco de Dados do Azure para PostgreSQL – grupo de servidores de Hiperescala (Citus) usando o portal do Azure. Você explorará os dados distribuídos: fragmentar tabelas entre os nós, ingerir dados de amostra e efetuar consultas que são executadas em vários nós.

Criar um grupo de servidores Hyperscale (Citus)

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Entre no Portal do Azure

Entre no [portal do Azure](#).

Siga estas etapas para criar um Banco de Dados do Azure para o servidor PostgreSQL:

1. Clique em **Criar um recurso** no canto superior esquerdo do portal do Azure.
2. Selecione **Bancos de Dados** na página **Novo** e selecione **Banco de Dados do Azure para PostgreSQL** na página **Bancos de Dados**.
3. Para a opção de implantação, clique no botão **Criar em grupo de servidores Hyperscale (Citus)**.
4. Preencha o formulário de detalhes sobre o novo servidor com as seguintes informações:
 - Grupo de recursos: clique no link **Criar novo** abaixo da caixa de texto desse campo. Insira um nome como **myresourcegroup**.
 - Nome do grupo de servidores: digite um nome exclusivo para o novo grupo de servidores que também poderá ser usado para um subdomínio do servidor.
 - Nome de usuário do administrador: no momento deve ser o valor **citus** e não pode ser alterado.
 - Senha: deve ter pelo menos oito caracteres de comprimento e conter caracteres das três seguintes categorias: letras maiúsculas em inglês, letras minúsculas em inglês, números (0 – 9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
 - Localização: use a localização mais próxima dos usuários para fornecer a eles acesso mais rápido aos dados.

IMPORTANT

A senha de administrador do servidor que você especificar aqui é necessária para fazer logon no servidor e nos bancos de dados. Lembre-se ou registre essas informações para o uso posterior.

5. Clique em **Configurar grupo de servidores**. Deixe as configurações nessa seção inalteradas e clique em **Salvar**.
6. Clique em **Próximo: Rede** > na parte inferior da tela.
7. Na guia **Rede**, clique no botão de opção **Ponto de extremidade público**.

Hyperscale (Citus) server group

Microsoft

Basics Networking Tags Review + create

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method i

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group i

No

Yes

[+ Add current client IP address \(207.153.12.62 \)](#) [+ Add 0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP
<input type="text"/>	<input type="text"/>	<input type="text"/>

8. Clique no link **+ Adicionar endereço IP do cliente atual**.

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method [\(i\)](#)

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group [\(i\)](#)

No Yes

+ Add current client IP address (207.153.12.62) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP	
ClientIPAddress_2020-7-27_18-50-22	207.153.12.62	207.153.12.62	
Firewall rule name	Start IP	End IP	

NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Se você estiver tentando se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Se isso acontecer, você não poderá se conectar ao cluster do Hiperescala (Citus), a menos que o departamento de TI abra a porta 5432.

9. Clique em **Examinar + criar** e, em seguida, **Criar** para provisionar o servidor. O provisionamento demora alguns minutos.
10. A página será redirecionada para monitorar a implantação. Quando o status em tempo real mudar de **Sua implantação está em andamento** para **Sua implantação está concluída**, clique no item de menu **Saídas** no lado esquerdo da página.
11. A página de saídas conterá um nome do host do coordenador com um botão ao lado dele para copiar o valor para a área de transferência. Registre essas informações para uso posterior.

Conectar-se ao banco de dados usando psql

Ao criar o servidor do Banco de Dados do Azure para PostgreSQL, um banco de dados padrão chamado **citus** é criado. Para se conectar ao seu servidor de banco de dados, é necessário uma cadeia de conexão e a senha de administrador.

1. Obter a cadeia de conexão. Na página de grupo de servidor, clique no item de menu **Cadeias de conexão**. (Está em **Configurações**.) Localize a cadeia de caracteres marcada como **psql**. Ela terá o formato:

```
psql "host=hostname.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Copie a cadeia de caracteres. Você precisará substituir "{sua_senha}" pela senha administrativa escolhida anteriormente. O sistema não armazena a senha de texto sem formatação e, portanto, não é possível exibi-la na cadeia de conexão.

2. Abra uma janela de terminal no computador local.
3. No prompt, conecte-se ao servidor do Banco de Dados do Azure para PostgreSQL com o utilitário [psql](#). Passe a cadeia de conexão entre aspas, certificando-se de que contém a senha:

```
psql "host=..."
```

Por exemplo, o comando a seguir conecta-se ao nó coordenador do grupo de servidores **mydemoserver**:

```
psql "host=mydemoserver-c.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Criar e distribuir tabelas

Uma vez conectado ao nó coordenador em hiperescala usando o psql, você pode concluir algumas tarefas básicas.

Dentro dos servidores da Hiperescala (Citus), há três tipos de tabelas:

- Tabelas fragmentadas ou distribuídas (espalhadas para auxiliar na escala de desempenho e paralelização)
- Tabelas de referência (várias cópias mantidas)
- Tabelas locais (geralmente usadas para tabelas de administrador interno)

Neste guia de início rápido, nos concentraremos principalmente em tabelas distribuídas e nos familiarizaremos com elas.

O modelo de dados que vamos trabalhar é simples: dados de evento e do usuário do GitHub. Eventos incluem a criação de fork, confirmações do git relacionadas a uma organização e muito mais.

Depois de se conectar via psql, vamos criar nossas tabelas. No console do psql, execute:

```

CREATE TABLE github_events
(
    event_id bigint,
    event_type text,
    event_public boolean,
    repo_id bigint,
    payload jsonb,
    repo jsonb,
    user_id bigint,
    org jsonb,
    created_at timestamp
);

CREATE TABLE github_users
(
    user_id bigint,
    url text,
    login text,
    avatar_url text,
    gravatar_id text,
    display_login text
);

```

O `payload` campo de `github_events` tem um tipo de dados JSONB. O JSONB é o tipo de dados JSON em formato binário no Postgres. O tipo de dados torna mais fácil armazenar um esquema flexível em uma única coluna.

O Postgres pode criar um índice `GIN` neste tipo que indexará cada chave e valor dentro dele. Com um índice, se tornará rápido e fácil consultar o conteúdo com várias condições. Vamos em frente para criar alguns índices antes de podermos carregar nossos dados. No psql:

```

CREATE INDEX event_type_index ON github_events (event_type);
CREATE INDEX payload_index ON github_events USING GIN (payload jsonb_path_ops);

```

Em seguida, vamos pegar as tabelas Postgres no nó coordenador e instruir a Hiperescala (Citus) a fragmentá-las entre os trabalhadores. Para fazer isso, executaremos uma consulta para cada tabela especificando a chave para fragmentá-la. No exemplo atual, fragmentaremos a tabela de eventos e de usuários em `user_id`:

```

SELECT create_distributed_table('github_events', 'user_id');
SELECT create_distributed_table('github_users', 'user_id');

```

IMPORTANT

A distribuição de tabelas é necessária para tirar proveito dos recursos de desempenho de Hiperescala. Se você não distribuir tabelas, os nós de trabalho não poderão ajudar a executar consultas que envolvam essas tabelas.

Estamos prontos para carregar dados. Ainda no psql, saia da shell para baixar os arquivos:

```

\! curl -O https://examples.citusdata.com/users.csv
\! curl -O https://examples.citusdata.com/events.csv

```

Em seguida, carregue os dados dos arquivos nas tabelas distribuídas:

```
SET CLIENT_ENCODING TO 'utf8';

\copy github_events from 'events.csv' WITH CSV
\copy github_users from 'users.csv' WITH CSV
```

Executar consultas

Agora é a hora da diversão: realmente efetuar algumas consultas. Vamos começar com um simples `count (*)` para ver a quantidade de dados carregada:

```
SELECT count(*) from github_events;
```

Funcionou muito bem. Voltaremos para esse tipo de agregação em breve, mas agora vamos examinar algumas outras consultas. Na coluna `payload` JSONB há um bom volume de dados, mas varia com base no tipo de evento. Os eventos `PushEvent` contêm um tamanho que inclui o número de confirmações distintos para o envio por push. É possível usar isso para localizar o número total de confirmações por hora:

```
SELECT date_trunc('hour', created_at) AS hour,
       sum((payload->>'distinct_size')::int) AS num_commits
  FROM github_events
 WHERE event_type = 'PushEvent'
 GROUP BY hour
 ORDER BY hour;
```

Até agora, as consultas envolveram os eventos `github` de modo exclusivo, mas podemos combinar essas informações com os usuários do `github`. Uma vez que criamos usuários fragmentados e eventos no mesmo identificador (`user_id`), as linhas de ambas as tabelas com IDs de usuário correspondentes serão **colocadas** nos mesmos nós de banco de dados e podem ser unidas facilmente.

Se ingressarmos em `user_id`, a Hiperescala (Citus) poderá efetuar push da execução da junção para os fragmentos para execução em paralelo nos nós de trabalho. Por exemplo, vamos encontrar os usuários que criaram o maior número de repositórios:

```
SELECT gu.login, count(*)
  FROM github_events ge
  JOIN github_users gu
    ON ge.user_id = gu.user_id
 WHERE ge.event_type = 'CreateEvent'
   AND ge.payload @> '{"ref_type": "repository"}'
 GROUP BY gu.login
 ORDER BY count(*) DESC;
```

Limpar os recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão **Excluir** na página **Visão geral** do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão **Excluir** final.

Próximas etapas

Neste guia de início rápido, você aprendeu como provisionar um grupo de servidores Hyperscale (Citus). Você conectou ele com o psql, criou um esquema e distribuiu dados.

- Siga um tutorial para [compilar aplicativos escalonáveis de multilocatário](#)
- Determinar o melhor [tamanho inicial](#) para seu grupo de servidores

Tutorial: criar grupo de servidores

21/05/2021 • 3 minutes to read

Neste tutorial, você cria um grupo de servidores no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus). Você executará estas etapas:

- Provisionar os nós
- Permitir acesso à rede
- Conectar-se ao nó coordenador

Criar um grupo de servidores Hyperscale (Citus)

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Entre no Portal do Azure

Entre no [portal do Azure](#).

Siga estas etapas para criar um Banco de Dados do Azure para o servidor PostgreSQL:

1. Clique em **Criar um recurso** no canto superior esquerdo do portal do Azure.
2. Selecione **Bancos de Dados** na página **Novo** e selecione **Banco de Dados do Azure para PostgreSQL** na página **Bancos de Dados**.
3. Para a opção de implantação, clique no botão **Criar em grupo de servidores Hyperscale (Citus)**.
4. Preencha o formulário de detalhes sobre o novo servidor com as seguintes informações:
 - Grupo de recursos: clique no link **Criar novo** abaixo da caixa de texto desse campo. Insira um nome como **myresourcegroup**.
 - Nome do grupo de servidores: digite um nome exclusivo para o novo grupo de servidores que também poderá ser usado para um subdomínio do servidor.
 - Nome de usuário do administrador: no momento deve ser o valor **citus** e não pode ser alterado.
 - Senha: deve ter pelo menos oito caracteres de comprimento e conter caracteres das três seguintes categorias: letras maiúsculas em inglês, letras minúsculas em inglês, números (0 – 9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
 - Localização: use a localização mais próxima dos usuários para fornecer a eles acesso mais rápido aos dados.

IMPORTANT

A senha de administrador do servidor que você especificar aqui é necessária para fazer logon no servidor e nos bancos de dados. Lembre-se ou registre essas informações para o uso posterior.

5. Clique em **Configurar grupo de servidores**. Deixe as configurações nessa seção inalteradas e clique em **Salvar**.
6. Clique em **Próximo: Rede >** na parte inferior da tela.
7. Na guia **Rede**, clique no botão de opção **Ponto de extremidade público**.

Hyperscale (Citus) server group

Microsoft

Basics Networking Tags Review + create

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method i

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group i

No

Yes

[+ Add current client IP address \(207.153.12.62 \)](#) [+ Add 0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP
<input type="text"/>	<input type="text"/>	<input type="text"/>

8. Clique no link **+ Adicionar endereço IP do cliente atual**.

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method [\(i\)](#)

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group [\(i\)](#)

No Yes

+ Add current client IP address (207.153.12.62) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP	
ClientIPAddress_2020-7-27_18-50-22	207.153.12.62	207.153.12.62	
Firewall rule name	Start IP	End IP	

NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Se você estiver tentando se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Se isso acontecer, você não poderá se conectar ao cluster do Hiperescala (Citus), a menos que o departamento de TI abra a porta 5432.

9. Clique em **Examinar + criar** e, em seguida, **Criar** para provisionar o servidor. O provisionamento demora alguns minutos.
10. A página será redirecionada para monitorar a implantação. Quando o status em tempo real mudar de **Sua implantação está em andamento** para **Sua implantação está concluída**, clique no item de menu **Saídas** no lado esquerdo da página.
11. A página de saídas conterá um nome do host do coordenador com um botão ao lado dele para copiar o valor para a área de transferência. Registre essas informações para uso posterior.

Conectar-se ao banco de dados usando psql

Ao criar o servidor do Banco de Dados do Azure para PostgreSQL, um banco de dados padrão chamado **citus** é criado. Para se conectar ao seu servidor de banco de dados, é necessário uma cadeia de conexão e a senha de administrador.

1. Obter a cadeia de conexão. Na página de grupo de servidor, clique no item de menu **Cadeias de conexão**. (Está em **Configurações**.) Localize a cadeia de caracteres marcada como **psql**. Ela terá o formato:

```
psql "host=hostname.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Copie a cadeia de caracteres. Você precisará substituir "{sua_senha}" pela senha administrativa escolhida anteriormente. O sistema não armazena a senha de texto sem formatação e, portanto, não é possível exibi-la na cadeia de conexão.

2. Abra uma janela de terminal no computador local.
3. No prompt, conecte-se ao servidor do Banco de Dados do Azure para PostgreSQL com o utilitário [psql](#). Passe a cadeia de conexão entre aspas, certificando-se de que contém a senha:

```
psql "host=..."
```

Por exemplo, o comando a seguir conecta-se ao nó coordenador do grupo de servidores **mydemoserver**:

```
psql "host=mydemoserver-c.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Próximas etapas

Com um grupo de servidores provisionado, é hora de passar para o próximo tutorial:

- [Trabalhar com os dados distribuídos](#)

Tutorial: Fragmentar dados em nós de trabalho no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

01/07/2021 • 7 minutes to read

Neste tutorial, você usa o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) para saber como:

- Criar fragmentos distribuídos por hash
- Ver o local em que os fragmentos de tabela são colocados
- Identificar distribuição distorcida
- Criar restrições em tabelas distribuídas
- Executar consultas em dados distribuídos

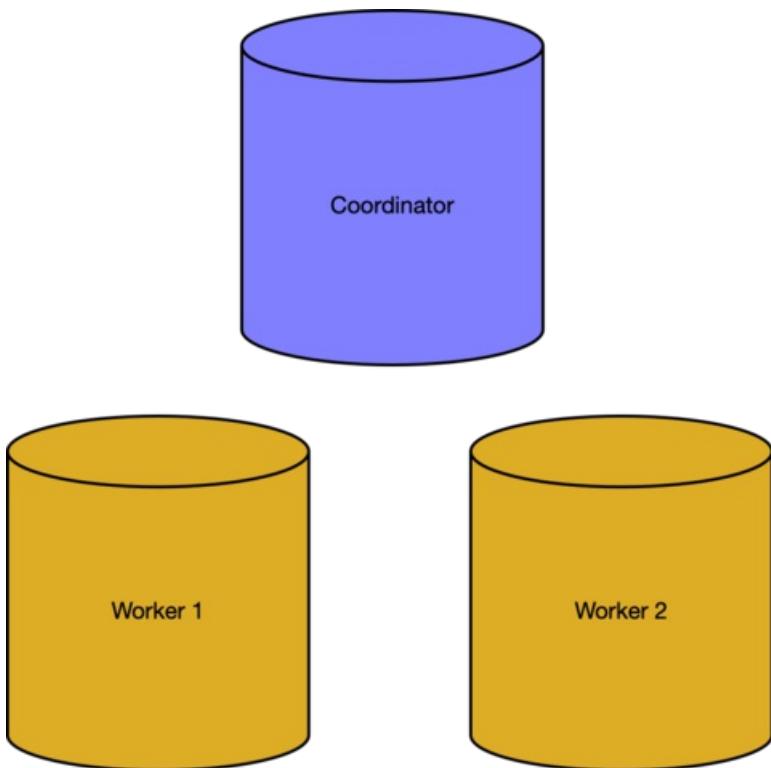
Pré-requisitos

Este tutorial requer um grupo de servidores Hiperescala (Citus) em execução com dois nós de trabalho. Se você não tiver um grupo de servidores em execução, siga o tutorial [criar grupo de servidores](#) e volte para este.

Dados distribuídos por hash

A distribuição de linhas de tabela em vários servidores PostgreSQL é uma técnica fundamental para consultas escalonáveis no Hiperescala (Citus). Juntos, vários nós podem conter mais dados do que um banco de dados tradicional e, em muitos casos, podem usar CPUs de trabalho em paralelo para executar consultas.

Na seção pré-requisitos, criamos um grupo de servidores Hiperescala (Citus) com dois nós de trabalho.



As tabelas de metadados do nó coordenador controlam os trabalhos e os dados distribuídos. Podemos verificar os trabalhos ativos na tabela [pg_dist_node](#).

```
select nodeid, nodename from pg_dist_node where isactive;
```

nodeid	nodename
1	10.0.0.21
2	10.0.0.23

NOTE

Os Nodenames no Hiperescala (Citus) são endereços IP internos em uma rede virtual e os endereços reais que você vê podem ser diferentes.

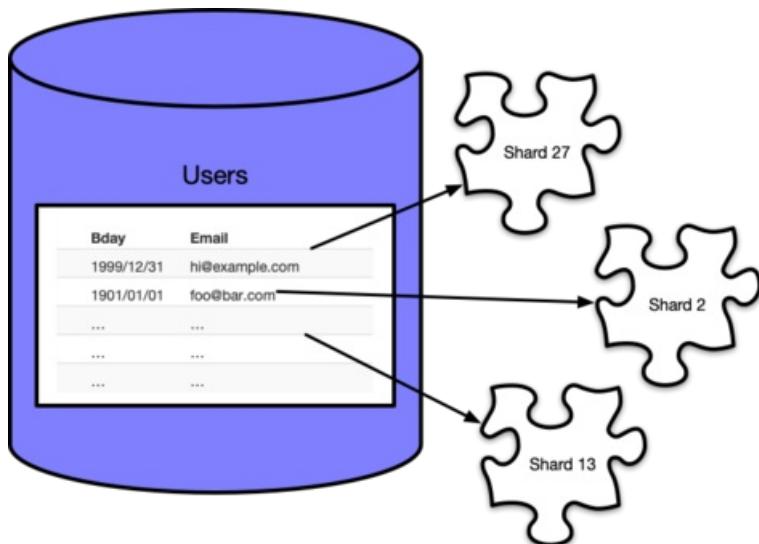
Linhas, fragmentos e posicionamentos

Para usar os recursos de CPU e de armazenamento dos nós de trabalho, precisamos distribuir dados de tabela em todo o grupo de servidores. A distribuição de uma tabela atribui cada linha a um grupo lógico chamado *fragmento*. Vamos criar uma tabela e distribuí-la:

```
-- create a table on the coordinator
create table users ( email text primary key, bday date not null );

-- distribute it into shards on workers
select create_distributed_table('users', 'email');
```

O Hiperescala (Citus) atribui cada linha a um fragmento com base no valor da *coluna de distribuição*, que, em nosso caso, especificamos como `email`. Cada linha estará em exatamente um fragmento e cada fragmento pode conter várias linhas.



Por padrão `create_distributed_table()` faz 32 fragmentos, como podemos ver contando na tabela de metadados `pg_dist_shard`:

```
select logicalrelid, count(shardid)
  from pg_dist_shard
 group by logicalrelid;
```

```

logicalrelid | count
-----+-----
users      |   32

```

O Hiperescala (Citus) usa a tabela `pg_dist_shard` para atribuir linhas a fragmentos, com base em um hash do valor na coluna de distribuição. Os detalhes do hash não são importantes para este tutorial. O que importa é que podemos consultar para ver quais valores são mapeados para quais IDs de fragmentos:

```

-- Where would a row containing hi@test.com be stored?
-- (The value doesn't have to actually be present in users, the mapping
-- is a mathematical operation consulting pg_dist_shard.)
select get_shard_id_for_distribution_column('users', 'hi@test.com');

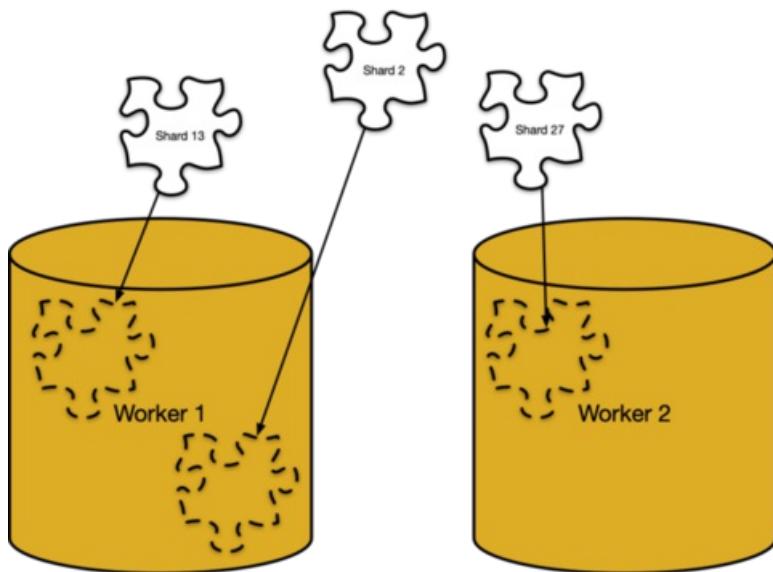
```

```

get_shard_id_for_distribution_column
-----
102008

```

O mapeamento de linhas para fragmentos é puramente lógico. Os fragmentos devem ser atribuídos a nós de trabalho específicos para armazenamento, o que o Hiperescala (Citus) chama de *posicionamento de fragmento*.



Podemos examinar os posicionamentos de fragmentos em [pg_dist_placement](#). Unindo-se a outras tabelas de metadados que vimos mostra o local em que cada fragmento reside.

```

-- limit the output to the first five placements

select
    shard.logicalrelid as table,
    placement.shardid as shard,
    node.nodename as host
from
    pg_dist_placement placement,
    pg_dist_node node,
    pg_dist_shard shard
where placement.groupid = node.groupid
    and shard.shardid = placement.shardid
order by shard
limit 5;

```

table	shard	host
users	102008	10.0.0.21
users	102009	10.0.0.23
users	102010	10.0.0.21
users	102011	10.0.0.23
users	102012	10.0.0.21

Distorção de dados

Um grupo de servidores é executado com mais eficiência quando você coloca os dados uniformemente em nós de trabalho e quando coloca dados relacionados juntos nos mesmos trabalhos. Nesta seção, vamos nos concentrar na primeira parte: a uniformidade do posicionamento.

Para demonstrar, vamos criar dados de exemplo para nossa tabela `users`:

```
-- load sample data
insert into users
select
    md5(random()::text) || '@test.com',
    date_trunc('day', now() - random() *'100 years'::interval)
from generate_series(1, 1000);
```

Para ver os tamanhos de fragmentos, podemos executar [funções de tamanho de tabela](#) nos fragmentos.

```
-- sizes of the first five shards
select *
from
    run_command_on_shards('users', $cmd$,
        select pg_size.pretty(pg_table_size('%1$s'));
    $cmd$)
order by shardid
limit 5;
```

shardid	success	result
102008	t	16 kB
102009	t	16 kB
102010	t	16 kB
102011	t	16 kB
102012	t	16 kB

Podemos ver que os fragmentos são de tamanho igual. Já vimos que os posicionamentos estão distribuídos uniformemente entre os trabalhadores, portanto, podemos inferir que os nós de trabalho contêm números de linhas aproximadamente iguais.

As linhas em nosso exemplo `users` estão distribuídas uniformemente devido às propriedades da coluna de distribuição, `email`.

1. O número de endereços de email era superior ou igual ao número de fragmentos.
2. O número de linhas por endereço de email era semelhante (em nosso caso, exatamente uma linha por endereço, porque declaramos o email como uma chave).

Qualquer opção de tabela e coluna de distribuição em que uma das propriedades falhar terminará com tamanho de dados irregular nos trabalhos, ou seja, *distorção de dados*.

Adicionar restrições a dados distribuídos

Usar o Hiperescala (Citus) permite que você continue a desfrutar da segurança de um banco de dados

relacional, incluindo [restrições de banco de dados](#). No entanto, há uma limitação. Por causa da natureza dos sistemas distribuídos, o Hiperescala (Citus) não faz referência a restrições de exclusividade ou integridade referencial entre nós de trabalho.

Vamos considerar nosso exemplo de tabela `users` com uma tabela relacionada.

```
-- books that users own
create table books (
    owner_email text references users (email),
    isbn text not null,
    title text not null
);

-- distribute it
select create_distributed_table('books', 'owner_email');
```

Por questão de eficiência, distribuímos `books` da mesma maneira que `users`: pelo endereço de email do proprietário. A distribuição por valores de coluna semelhantes é chamada de [colocação](#).

Não tínhamos problema ao distribuir livros com uma chave estrangeira para os usuários, pois a chave estava em uma coluna de distribuição. No entanto, teríamos problemas para tornar `isbn` uma chave:

```
-- will not work
alter table books add constraint books_isbn unique (isbn);
```

```
ERROR:  cannot create constraint on "books"
DETAIL: Distributed relations cannot have UNIQUE, EXCLUDE, or
        PRIMARY KEY constraints that do not include the partition column
        (with an equality operator if EXCLUDE).
```

Em uma tabela distribuída, o melhor que podemos fazer é tornar as colunas um módulo exclusivo da coluna de distribuição:

```
-- a weaker constraint is allowed
alter table books add constraint books_isbn unique (owner_email, isbn);
```

A restrição acima simplesmente torna o ISBN exclusivo por usuário. Outra opção é tornar os livros uma [tabela de referência](#) em vez de uma tabela distribuída e criar uma tabela distribuída separada associando livros a usuários.

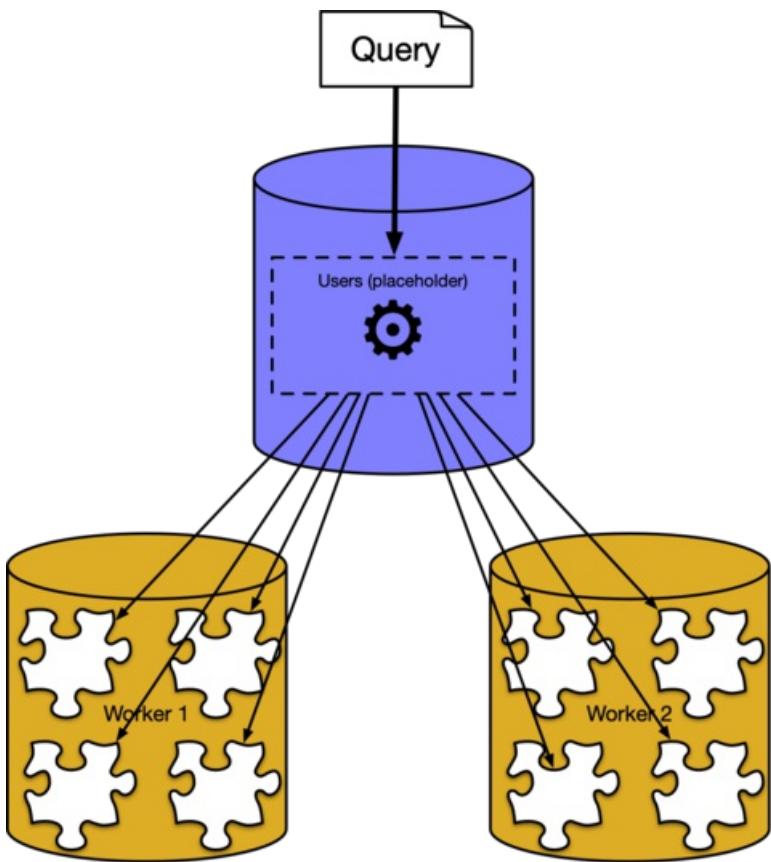
Consultar tabelas distribuídas

Nas seções anteriores, vimos como as linhas da tabela distribuída são colocadas em fragmentos em nós de trabalho. Na maioria das vezes, você não precisa saber como ou o local em que os dados são armazenados em um grupo de servidores. O Hiperescala (Citus) tem um executor de consulta distribuída que divide automaticamente consultas SQL regulares. Ele as executa em paralelo em nós de trabalho próximos aos dados.

Por exemplo, podemos executar uma consulta para encontrar a idade média dos usuários, tratando a tabela `users` distribuída como uma tabela normal no coordenador.

```
select avg(current_date - bday) as avg_days_old from users;
```

```
avg_days_old  
-----  
17926.348000000000
```



Nos bastidores, o executor do Hiperescala (Citus) cria uma consulta separada para cada fragmento, executa-as nos trabalhos e combina o resultado. Veja isso usando o comando EXPLAIN do PostgreSQL:

```
explain select avg(current_date - bday) from users;
```

```
QUERY PLAN  
-----  
Aggregate (cost=500.00..500.02 rows=1 width=32)  
  -> Custom Scan (Citus Adaptive) (cost=0.00..0.00 rows=100000 width=16)  
    Task Count: 32  
    Tasks Shown: One of 32  
      -> Task  
        Node: host=10.0.0.21 port=5432 dbname=citus  
          -> Aggregate (cost=41.75..41.76 rows=1 width=16)  
            -> Seq Scan on users_102040 users (cost=0.00..22.70 rows=1270 width=4)
```

A saída mostra um exemplo de um plano de execução para um *fragmento de consulta* em execução no fragmento 102040 (a tabela `users_102040` no trabalho 10.0.0.21). Os outros fragmentos não são mostrados porque são semelhantes. Podemos ver que o nó de trabalho examina as tabelas de fragmentos e aplica a agregação. O nó coordenador combina as agregações para o resultado final.

Próximas etapas

Neste tutorial, criamos uma tabela distribuída e aprendemos sobre seus fragmentos e posicionamentos. Vimos um desafio de usar restrições de exclusividade e de chave estrangeira e, finalmente, vimos como as consultas distribuídas funcionam em um alto nível.

- Leia mais sobre [tipos de tabela](#) do Hiperescala (Citus)
- Obtenha mais dicas sobre [como escolher uma coluna de distribuição](#)
- Conheça os benefícios da [colocação de tabela](#)

Tutorial: criar um banco de dados multilocatário usando o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 9 minutes to read

Neste tutorial, você usa o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) para saber como:

- Criar um grupo de servidores Hyperscale (Citus)
- Usar o utilitário psql para criar um esquema
- Fragmentar tabelas entre nós
- Ingerir dados de exemplo
- Consultar dados do locatário
- Consultar dados entre locatários
- Personalizar o esquema por locatário

Pré-requisitos

Criar um grupo de servidores Hyperscale (Citus)

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Entre no Portal do Azure

Entre no [portal do Azure](#).

Siga estas etapas para criar um Banco de Dados do Azure para o servidor PostgreSQL:

1. Clique em **Criar um recurso** no canto superior esquerdo do portal do Azure.
2. Selecione **Bancos de Dados** na página **Novo** e selecione **Banco de Dados do Azure para PostgreSQL** na página **Bancos de Dados**.
3. Para a opção de implantação, clique no botão **Criar em grupo de servidores Hyperscale (Citus)**.
4. Preencha o formulário de detalhes sobre o novo servidor com as seguintes informações:
 - Grupo de recursos: clique no link **Criar novo** abaixo da caixa de texto desse campo. Insira um nome como **myresourcegroup**.
 - Nome do grupo de servidores: digite um nome exclusivo para o novo grupo de servidores que também poderá ser usado para um subdomínio do servidor.
 - Nome de usuário do administrador: no momento deve ser o valor **citus** e não pode ser alterado.
 - Senha: deve ter pelo menos oito caracteres de comprimento e conter caracteres das três seguintes categorias: letras maiúsculas em inglês, letras minúsculas em inglês, números (0 – 9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
 - Localização: use a localização mais próxima dos usuários para fornecer a eles acesso mais rápido aos dados.

IMPORTANT

A senha de administrador do servidor que você especificar aqui é necessária para fazer logon no servidor e nos bancos de dados. Lembre-se ou registre essas informações para o uso posterior.

5. Clique em **Configurar grupo de servidores**. Deixe as configurações nessa seção inalteradas e clique em **Salvar**.
6. Clique em **Próximo: Rede** > na parte inferior da tela.
7. Na guia **Rede**, clique no botão de opção **Ponto de extremidade público**.

Hyperscale (Citus) server group

Microsoft

Basics **Networking** Tags Review + create

Configure networking access and security for your server group. [Learn more](#) 

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method 

No access Public endpoint

 Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group 

No Yes

[+ Add current client IP address \(207.153.12.62 \)](#) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP
<input type="text"/>	<input type="text"/>	<input type="text"/>

8. Clique no link **+ Adicionar endereço IP do cliente atual**.

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method [\(i\)](#)

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group [\(i\)](#)

No Yes

+ Add current client IP address (207.153.12.62) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP	
ClientIPAddress_2020-7-27_18-50-22	207.153.12.62	207.153.12.62	
Firewall rule name	Start IP	End IP	

NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Se você estiver tentando se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Se isso acontecer, você não poderá se conectar ao cluster do Hiperescala (Citus), a menos que o departamento de TI abra a porta 5432.

9. Clique em **Examinar + criar** e, em seguida, **Criar** para provisionar o servidor. O provisionamento demora alguns minutos.
10. A página será redirecionada para monitorar a implantação. Quando o status em tempo real mudar de **Sua implantação está em andamento** para **Sua implantação está concluída**, clique no item de menu **Saídas** no lado esquerdo da página.
11. A página de saídas conterá um nome do host do coordenador com um botão ao lado dele para copiar o valor para a área de transferência. Registre essas informações para uso posterior.

Conectar-se ao banco de dados usando psql

Ao criar o servidor do Banco de Dados do Azure para PostgreSQL, um banco de dados padrão chamado **citus** é criado. Para se conectar ao seu servidor de banco de dados, é necessário uma cadeia de conexão e a senha de administrador.

1. Obter a cadeia de conexão. Na página de grupo de servidor, clique no item de menu **Cadeias de conexão**. (Está em **Configurações**.) Localize a cadeia de caracteres marcada como **psql**. Ela terá o formato:

```
psql "host=hostname.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Copie a cadeia de caracteres. Você precisará substituir "{sua_senha}" pela senha administrativa escolhida anteriormente. O sistema não armazena a senha de texto sem formatação e, portanto, não é possível exibi-la na cadeia de conexão.

2. Abra uma janela de terminal no computador local.
3. No prompt, conecte-se ao servidor do Banco de Dados do Azure para PostgreSQL com o utilitário [psql](#). Passe a cadeia de conexão entre aspas, certificando-se de que contém a senha:

```
psql "host=..."
```

Por exemplo, o comando a seguir conecta-se ao nó coordenador do grupo de servidores **mydemoserver**:

```
psql "host=mydemoserver-c.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Usar o utilitário psql para criar um esquema

Uma vez conectado ao Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) usando o psql, é possível realizar algumas tarefas básicas. Este tutorial orienta você durante a criação de um aplicativo Web que permite aos anunciantes acompanharem suas campanhas.

Várias empresas podem usar o aplicativo, então, vamos criar uma tabela para armazenar as empresas e outra para as campanhas delas. No console do psql, execute estes comandos:

```
CREATE TABLE companies (
    id bigserial PRIMARY KEY,
    name text NOT NULL,
    image_url text,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL
);

CREATE TABLE campaigns (
    id bigserial,
    company_id bigint REFERENCES companies (id),
    name text NOT NULL,
    cost_model text NOT NULL,
    state text NOT NULL,
    monthly_budget bigint,
    blacklisted_site_urls text[],
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,
    PRIMARY KEY (company_id, id)
);
```

NOTE

Este artigo contém referências ao termo *não autorizado*, um termo que a Microsoft não usa mais. Quando o termo for removido do software, também o removeremos deste artigo.

Cada campanha pagará para executar anúncios. Também adicione uma tabela para anúncios, executando o seguinte código no psql após o código acima:

```
CREATE TABLE ads (
    id bigserial,
    company_id bigint,
    campaign_id bigint,
    name text NOT NULL,
    image_url text,
    target_url text,
    impressions_count bigint DEFAULT 0,
    clicks_count bigint DEFAULT 0,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,
    PRIMARY KEY (company_id, id),
    FOREIGN KEY (company_id, campaign_id)
        REFERENCES campaigns (company_id, id)
);
```

Por fim, vamos rastrear as estatísticas sobre cliques e impressões para cada anúncio:

```
CREATE TABLE clicks (
    id bigserial,
    company_id bigint,
    ad_id bigint,
    clicked_at timestamp without time zone NOT NULL,
    site_url text NOT NULL,
    cost_per_click_usd numeric(20,10),
    user_ip inet NOT NULL,
    user_data jsonb NOT NULL,
    PRIMARY KEY (company_id, id),
    FOREIGN KEY (company_id, ad_id)
        REFERENCES ads (company_id, id)
);

CREATE TABLE impressions (
    id bigserial,
    company_id bigint,
    ad_id bigint,
    seen_at timestamp without time zone NOT NULL,
    site_url text NOT NULL,
    cost_per_impression_usd numeric(20,10),
    user_ip inet NOT NULL,
    user_data jsonb NOT NULL,
    PRIMARY KEY (company_id, id),
    FOREIGN KEY (company_id, ad_id)
        REFERENCES ads (company_id, id)
);
```

Agora é possível ver as tabelas recém-criadas na lista de tabelas com este comando do psql:

```
\dt
```

Aplicativos multilocatários podem impor exclusividade apenas por locatário, por isso, todas as chaves primárias e estrangeiras incluem a ID da empresa.

Fragmentar tabelas entre nós

Uma implantação em hiperescala armazena as linhas da tabela em diferentes nós com base no valor de uma coluna designada pelo usuário. Essa "coluna de distribuição" marca qual locatário possui quais linhas.

Vamos definir a coluna de distribuição como empresa_id, o identificador do locatário. No psql, execute estas funções:

```
SELECT create_distributed_table('companies',      'id');
SELECT create_distributed_table('campaigns',       'company_id');
SELECT create_distributed_table('ads',              'company_id');
SELECT create_distributed_table('clicks',           'company_id');
SELECT create_distributed_table('impressions',     'company_id');
```

IMPORTANT

A distribuição de tabelas é necessária para tirar proveito dos recursos de desempenho de Hiperescala. Se você não distribuir tabelas, os nós de trabalho não poderão ajudar a executar consultas que envolvam essas tabelas.

Ingerir dados de exemplo

Agora, fora do psql, na linha de comando normal, baixe conjuntos de dados de exemplo:

```
for dataset in companies campaigns ads clicks impressions geo_ips; do
    curl -O https://examples.citusdata.com/mt_ref_arch/${dataset}.csv
done
```

Novamente dentro do psql, carregue em massa os dados. Certifique-se de executar o psql no mesmo diretório em que você baixou os arquivos de dados.

```
SET CLIENT_ENCODING TO 'utf8';

\copy companies from 'companies.csv' with csv
\copy campaigns from 'campaigns.csv' with csv
\copy ads from 'ads.csv' with csv
\copy clicks from 'clicks.csv' with csv
\copy impressions from 'impressions.csv' with csv
```

Esses dados agora serão espalhados entre nós de trabalho.

Consultar dados do locatário

Quando o aplicativo solicita dados para um locatário único, o banco de dados pode executar a consulta em um nó de trabalho único. Consultas de locatário único filtram por uma ID de locatário único. Por exemplo, a consulta a seguir filtra `company_id = 5` para anúncios e impressões. Tente executá-la no psql para ver os resultados.

```

SELECT a.campaign_id,
       RANK() OVER (
           PARTITION BY a.campaign_id
           ORDER BY a.campaign_id, count(*) desc
       ), count(*) as n_impressions, a.id
  FROM ads as a
 JOIN impressions as i
   ON i.company_id = a.company_id
  AND i.ad_id      = a.id
 WHERE a.company_id = 5
GROUP BY a.campaign_id, a.id
ORDER BY a.campaign_id, n_impressions desc;

```

Consultar dados entre locatários

Até agora todas as tabelas foram distribuídas por `company_id`, mas alguns dados não "pertencem" naturalmente a qualquer locatário em particular e podem ser compartilhados. Por exemplo, todas as empresas na plataforma de anúncios de exemplo talvez queiram obter informações geográficas do respectivo público-alvo com base em endereços IP.

Crie uma tabela para armazenar informações geográficas compartilhadas. Executar os seguintes comandos na psql:

```

CREATE TABLE geo_ips (
    addrs cidr NOT NULL PRIMARY KEY,
    latlon point NOT NULL
        CHECK (-90 <= latlon[0] AND latlon[0] <= 90 AND
              -180 <= latlon[1] AND latlon[1] <= 180)
);
CREATE INDEX ON geo_ips USING gist (addrs inet_ops);

```

Em seguida, faça com `geo_ips` uma "tabela de referência" para armazenar uma cópia da tabela em cada nó de trabalho.

```
SELECT create_reference_table('geo_ips');
```

Carregue-a com dados de exemplo. Lembre-se de executar esse comando em psql de dentro do diretório em que o conjunto de dados foi baixado.

```
\copy geo_ips from 'geo_ips.csv' with csv
```

Unir a tabela de cliques com `geo_ips` é eficiente em todos os nós. Aqui está uma junção para encontrar as localizações de todos aqueles que clicaram no anúncio 290. Tente executar a consulta no psql.

```

SELECT c.id, clicked_at, latlon
  FROM geo_ips, clicks c
 WHERE addrs >> c.user_ip
   AND c.company_id = 5
   AND c.ad_id = 290;

```

Personalizar o esquema por locatário

Cada locatário pode precisar armazenar informações especiais não necessárias para outras pessoas. No entanto, todos os locatários compartilham uma infraestrutura comum com um esquema de banco de dados idêntico.

Para onde os dados extras podem ir?

Um truque é usar um tipo de coluna aberta como JSONB do PostgreSQL. Nossa esquema tem um campo JSONB `clicks` chamado `user_data`. Uma empresa (digamos empresa cinco), pode usar a coluna para rastrear se o usuário está em um dispositivo móvel.

Aqui está uma consulta para descobrir quem clica mais: visitantes móveis ou tradicionais.

```
SELECT
    user_data->>'is_mobile' AS is_mobile,
    count(*) AS count
FROM clicks
WHERE company_id = 5
GROUP BY user_data->>'is_mobile'
ORDER BY count DESC;
```

Podemos otimizar essa consulta para uma única empresa criando um [índice parcial](#).

```
CREATE INDEX click_user_data_is_mobile
ON clicks ((user_data->>'is_mobile'))
WHERE company_id = 5;
```

De modo mais geral, é possível criar [índices GIN](#) em cada chave e valor dentro da coluna.

```
CREATE INDEX click_user_data
ON clicks USING gin (user_data);

-- this speeds up queries like, "which clicks have
-- the is_mobile key present in user_data?"

SELECT id
  FROM clicks
 WHERE user_data ? 'is_mobile'
   AND company_id = 5;
```

Limpar os recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão *Excluir* na página *Visão geral* do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão *Excluir* final.

Próximas etapas

Neste tutorial, você aprendeu a provisionar um grupo de servidores Hyperscale (Citus). Você conectou ele com o psql, criou um esquema e distribuiu dados. Você aprendeu a consultar dados de dentro e entre locatários, além de personalizar o esquema por locatário.

- Saiba mais sobre os [tipos de nó](#) do grupo de servidores
- Determinar o melhor [tamanho inicial](#) para seu grupo de servidores

Tutorial: Criar um painel de análise em tempo real usando o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 8 minutes to read

Neste tutorial, você usa o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) para saber como:

- Criar um grupo de servidores Hyperscale (Citus)
- Usar o utilitário psql para criar um esquema
- Fragmentar tabelas entre nós
- Gerar dados de exemplo
- Executar rollups
- Consultar dados brutos e agregados
- Expirar dados

Pré-requisitos

Criar um grupo de servidores Hyperscale (Citus)

Se você não tiver uma assinatura do Azure, crie uma conta [gratuita](#) antes de começar.

Entre no Portal do Azure

Entre no [portal do Azure](#).

Siga estas etapas para criar um Banco de Dados do Azure para o servidor PostgreSQL:

1. Clique em **Criar um recurso** no canto superior esquerdo do portal do Azure.
2. Selecione **Bancos de Dados** na página **Novo** e selecione **Banco de Dados do Azure para PostgreSQL** na página **Bancos de Dados**.
3. Para a opção de implantação, clique no botão **Criar em grupo de servidores Hyperscale (Citus)**.
4. Preencha o formulário de detalhes sobre o novo servidor com as seguintes informações:
 - Grupo de recursos: clique no link **Criar novo** abaixo da caixa de texto desse campo. Insira um nome como **myresourcegroup**.
 - Nome do grupo de servidores: digite um nome exclusivo para o novo grupo de servidores que também poderá ser usado para um subdomínio do servidor.
 - Nome de usuário do administrador: no momento deve ser o valor **citus** e não pode ser alterado.
 - Senha: deve ter pelo menos oito caracteres de comprimento e conter caracteres das três seguintes categorias: letras maiúsculas em inglês, letras minúsculas em inglês, números (0 – 9) e caracteres não alfanuméricos (!, \$, #, % e assim por diante).
 - Localização: use a localização mais próxima dos usuários para fornecer a eles acesso mais rápido aos dados.

IMPORTANT

A senha de administrador do servidor que você especificar aqui é necessária para fazer logon no servidor e nos bancos de dados. Lembre-se ou registre essas informações para o uso posterior.

5. Clique em **Configurar grupo de servidores**. Deixe as configurações nessa seção inalteradas e clique em **Salvar**.
6. Clique em **Próximo: Rede** > na parte inferior da tela.
7. Na guia **Rede**, clique no botão de opção **Ponto de extremidade público**.

Hyperscale (Citus) server group

Microsoft

Basics **Networking** Tags Review + create

Configure networking access and security for your server group. [Learn more](#) 

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method 

No access Public endpoint

 Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group 

No Yes

[+ Add current client IP address \(207.153.12.62 \)](#) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP
<input type="text"/>	<input type="text"/>	<input type="text"/>

8. Clique no link **+ Adicionar endereço IP do cliente atual**.

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method [\(i\)](#)

No access Public endpoint

i Connections from the IPs configured in the Firewall rules below will have access to this server group. By default, no public IPs are allowed.

Firewall rules

Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator in this server group.

Allow Azure services and resources to access this server group [\(i\)](#)

No Yes

+ Add current client IP address (207.153.12.62) [+ Add 0.0.0.0 - 255.255.255.255](#)

Firewall rule name	Start IP	End IP	
ClientIPAddress_2020-7-27_18-50-22	207.153.12.62	207.153.12.62	
Firewall rule name	Start IP	End IP	

NOTE

O servidor PostgreSQL do Azure se comunica pela porta 5432. Se você estiver tentando se conectar de dentro de uma rede corporativa, o tráfego de saída pela porta 5432 talvez não seja permitido pelo firewall de sua rede. Se isso acontecer, você não poderá se conectar ao cluster do Hiperescala (Citus), a menos que o departamento de TI abra a porta 5432.

- Clique em **Examinar + criar** e, em seguida, **Criar** para provisionar o servidor. O provisionamento demora alguns minutos.
- A página será redirecionada para monitorar a implantação. Quando o status em tempo real mudar de **Sua implantação está em andamento** para **Sua implantação está concluída**, clique no item de menu **Saídas** no lado esquerdo da página.
- A página de saídas conterá um nome do host do coordenador com um botão ao lado dele para copiar o valor para a área de transferência. Registre essas informações para uso posterior.

Conectar-se ao banco de dados usando psql

Ao criar o servidor do Banco de Dados do Azure para PostgreSQL, um banco de dados padrão chamado **citus** é criado. Para se conectar ao seu servidor de banco de dados, é necessário uma cadeia de conexão e a senha de administrador.

- Obter a cadeia de conexão. Na página de grupo de servidor, clique no item de menu **Cadeias de conexão**. (Está em **Configurações**.) Localize a cadeia de caracteres marcada como **psql**. Ela terá o formato:

```
psql "host=hostname.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Copie a cadeia de caracteres. Você precisará substituir "{sua_senha}" pela senha administrativa escolhida anteriormente. O sistema não armazena a senha de texto sem formatação e, portanto, não é possível exibi-la na cadeia de conexão.

2. Abra uma janela de terminal no computador local.
3. No prompt, conecte-se ao servidor do Banco de Dados do Azure para PostgreSQL com o utilitário [psql](#). Passe a cadeia de conexão entre aspas, certificando-se de que contém a senha:

```
psql "host=..."
```

Por exemplo, o comando a seguir conecta-se ao nó coordenador do grupo de servidores **mydemoserver**:

```
psql "host=mydemoserver-c.postgres.database.azure.com port=5432 dbname=citus user=citus password={your_password} sslmode=require"
```

Usar o utilitário psql para criar um esquema

Uma vez conectado ao Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) usando o psql, é possível realizar algumas tarefas básicas. Este tutorial orienta você a ingerir dados de tráfego do Web Analytics e, em seguida, acumular os dados para fornecer painéis em tempo real com base nesses dados.

Vamos criar uma tabela que consumirá todos os nossos dados brutos de tráfego da Web. Execute os seguintes comandos no terminal do psql:

```
CREATE TABLE http_request (
    site_id INT,
    ingest_time TIMESTAMPTZ DEFAULT now(),
    url TEXT,
    request_country TEXT,
    ip_address TEXT,
    status_code INT,
    response_time_msec INT
);
```

Também vamos criar uma tabela que conterá nossas agregações por minuto e uma tabela que mantém a posição do nosso último rollup. Execute os seguintes comandos em psql também:

```

CREATE TABLE http_request_1min (
    site_id INT,
    ingest_time TIMESTAMPTZ, -- which minute this row represents

    error_count INT,
    success_count INT,
    request_count INT,
    average_response_time_msec INT,
    CHECK (request_count = error_count + success_count),
    CHECK (ingest_time = date_trunc('minute', ingest_time))
);

CREATE INDEX http_request_1min_idx ON http_request_1min (site_id, ingest_time);

CREATE TABLE latest_rollup (
    minute timestamp PRIMARY KEY,

    CHECK (minute = date_trunc('minute', minute))
);

```

Agora é possível ver as tabelas recém-criadas na lista de tabelas com este comando do psql:

```
\dt
```

Fragmentar tabelas entre nós

Uma implantação em hiperescala armazena as linhas da tabela em diferentes nós com base no valor de uma coluna designada pelo usuário. Essa "coluna de distribuição" marca como os dados são fragmentados entre nós.

Vamos definir a coluna de distribuição como `site_id`, a chave de fragmentação. No psql, execute estas funções:

```

SELECT create_distributed_table('http_request',      'site_id');
SELECT create_distributed_table('http_request_1min', 'site_id');

```

IMPORTANT

A distribuição de tabelas é necessária para tirar proveito dos recursos de desempenho de Hiperescala. Se você não distribuir tabelas, os nós de trabalho não poderão ajudar a executar consultas que envolvam essas tabelas.

Gerar dados de exemplo

Agora, nosso grupo de servidores deve estar pronto para ingerir alguns dados. Podemos executar o seguinte localmente da nossa conexão `psql` para inserir dados continuamente.

```

DO $$

BEGIN LOOP
    INSERT INTO http_request (
        site_id, ingest_time, url, request_country,
        ip_address, status_code, response_time_msec
    ) VALUES (
        trunc(random()*32), clock_timestamp(),
        concat('http://example.com/', md5(random()::text)),
        ('{China,India,USA,Indonesia} '::text[])[ceil(random()*4)],
        concat(
            trunc(random()*250 + 2), '.',
            trunc(random()*250 + 2), '.',
            trunc(random()*250 + 2), '.',
            trunc(random()*250 + 2)
        )::inet,
        ('{200,404}' ::int[])[ceil(random()*2)],
        5+trunc(random()*150)
    );
    COMMIT;
    PERFORM pg_sleep(random() * 0.25);
END LOOP;
END $$;

```

A consulta insere aproximadamente oito linhas por segundo. As linhas são armazenadas em diferentes nós de trabalho, conforme direcionado pela coluna de distribuição, `site_id`.

NOTE

Deixe a consulta de geração de dados em execução e abra uma segunda conexão do psql para os comandos restantes neste tutorial.

Consulta

A opção de hospedagem em hiperescala permite que vários nós processem as consultas em paralelo para aumentar a velocidade. Por exemplo, o banco de dados calcula agregações como SUM e COUNT em nós de trabalho e combina os resultados em uma resposta final.

Aqui está uma consulta para contagem de solicitações da Web por minuto, juntamente com algumas estatísticas. Tente executá-la no psql e observe os resultados.

```

SELECT
    site_id,
    date_trunc('minute', ingest_time) as minute,
    COUNT(1) AS request_count,
    SUM(CASE WHEN (status_code between 200 and 299) THEN 1 ELSE 0 END) as success_count,
    SUM(CASE WHEN (status_code between 200 and 299) THEN 0 ELSE 1 END) as error_count,
    SUM(response_time_msec) / COUNT(1) AS average_response_time_msec
FROM http_request
WHERE date_trunc('minute', ingest_time) > now() - '5 minutes'::interval
GROUP BY site_id, minute
ORDER BY minute ASC;

```

Acumulando dados

A consulta anterior funciona bem nos estágios iniciais, mas o desempenho diminuirá conforme os dados são escalados. Até mesmo com o processamento distribuído, é mais rápido pré-computar os dados do que recalculá-los repetidamente.

Podemos garantir que nosso painel permaneça rápido, acumulando regularmente os dados brutos em uma tabela de agregação. Você pode experimentar com a duração de agregação. Usamos uma tabela de agregação por minuto, mas você poderia dividir dados em 5, 15 ou 60 minutos em vez disso.

Para executar esse pacote cumulativo mais facilmente, vamos colocá-lo em uma função plpgsql. Execute estes comandos no psql para criar a função `rollup_http_request`.

```
-- initialize to a time long ago
INSERT INTO latest_rollup VALUES ('10-10-1901');

-- function to do the rollup
CREATE OR REPLACE FUNCTION rollup_http_request() RETURNS void AS $$

DECLARE
    curr_rollup_time timestamp := date_trunc('minute', now());
    last_rollup_time timestamp := minute from latest_rollup;
BEGIN
    INSERT INTO http_request_1min (
        site_id, ingest_time, request_count,
        success_count, error_count, average_response_time_msec
    ) SELECT
        site_id,
        date_trunc('minute', ingest_time),
        COUNT(1) as request_count,
        SUM(CASE WHEN (status_code between 200 and 299) THEN 1 ELSE 0 END) as success_count,
        SUM(CASE WHEN (status_code between 200 and 299) THEN 0 ELSE 1 END) as error_count,
        SUM(response_time_msec) / COUNT(1) AS average_response_time_msec
    FROM http_request
    -- roll up only data new since last_rollup_time
    WHERE date_trunc('minute', ingest_time) <@
        tstzrange(last_rollup_time, curr_rollup_time, '[]')
    GROUP BY 1, 2;

    -- update the value in latest_rollup so that next time we run the
    -- rollup it will operate on data newer than curr_rollup_time
    UPDATE latest_rollup SET minute = curr_rollup_time;
END;
$$ LANGUAGE plpgsql;
```

Com nossa função no local, execute-a para acumular os dados:

```
SELECT rollup_http_request();
```

E com nossos dados em um formulário pré-agregado, é possível consultar a tabela de rollup para obter o mesmo relatório anterior. Execute a seguinte consulta:

```
SELECT site_id, ingest_time as minute, request_count,
       success_count, error_count, average_response_time_msec
  FROM http_request_1min
 WHERE ingest_time > date_trunc('minute', now()) - '5 minutes'::interval;
```

Expirar dados antigos

Os rollups realizam consultas mais rapidamente, porém ainda é necessário expirar os dados antigos para evitar custos de armazenamento não associados. Decida por quanto tempo você quer manter os dados para cada granularidade e usar consultas padrão para excluir dados expirados. No exemplo a seguir, decidimos manter os dados brutos de um dia e agregações por minuto de um mês:

```
DELETE FROM http_request WHERE ingest_time < now() - interval '1 day';
DELETE FROM http_request_1min WHERE ingest_time < now() - interval '1 month';
```

Em produção, é possível encapsular essas consultas em uma função e chamá-la a cada minuto em um trabalho de Cron.

Limpar os recursos

Nas etapas anteriores, você criou recursos do Azure em um grupo de servidores. Caso esses recursos não sejam mais necessários no futuro, exclua o grupo de servidores. Pressione o botão *Excluir* na página *Visão geral* do grupo de servidores. Quando solicitado em uma página pop-up, confirme o nome do grupo de servidores e clique no botão *Excluir* final.

Próximas etapas

Neste tutorial, você aprendeu a provisionar um grupo de servidores Hyperscale (Citus). Você conectou ele com o psql, criou um esquema e distribuiu dados. Você aprendeu a consultar os dados na forma bruta, agregar regularmente esses dados, consultar as tabelas agregadas e expirar os dados antigos.

- Saiba mais sobre os [tipos de nó](#) do grupo de servidores
- Determinar o melhor [tamanho inicial](#) para seu grupo de servidores

Camada básica (versão prévia)

21/05/2021 • 2 minutes to read

IMPORTANT

A camada básica de Hiperescala (Citus) está atualmente em versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

É possível ver a lista completa dos novos recursos em [Versão prévia dos recursos para Hiperescala \(Citus\)](#).

A camada básica no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) é uma maneira simples de criar um pequeno grupo de servidores que pode ser escalado posteriormente. Embora os grupos de servidores na camada standard tenham um nó de coordenador e pelo menos dois nós de trabalho, a camada básica executa tudo em um único nó de banco de dados.

Além de usar menos nós, a camada básica tem todos os recursos da camada standard. Assim como a camada standard, ela dá suporte à alta disponibilidade, réplicas de leitura e armazenamento de tabela de coluna, entre outros recursos.

Escolher a camada básica versus a standard

A camada básica pode ser uma opção de implantação econômica e conveniente para desenvolvimento inicial, teste e integração contínua. Ele usa um único nó de banco de dados e apresenta a mesma API do SQL que a camada standard. É possível testar aplicativos com a camada básica e, posteriormente, [graduar para a camada standard](#) com confiança de que a interface permanecerá a mesma.

A camada básica também é apropriada para cargas de trabalho menores em produção (quando sair da versão prévia para a disponibilidade geral). Há espaço para escalar verticalmente *dentro* da camada básica aumentando o número de vCores do servidor.

Quando é necessária uma escala maior de imediato, use a camada standard. O menor grupo de servidores permitido tem um nó de coordenador e dois trabalhadores. É possível optar por usar mais nós com base no caso de uso, conforme descrito em como fazer o [dimensionamento inicial](#).

Próximas etapas

- Saiba como [provisionar a camada básica](#)
- Quando estiver pronto, consulte [como graduar](#) da camada básica para a camada standard
- A opção de [armazenamento de colunas](#) está disponível nas camadas básica e standard

Nós e tabelas no Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)

21/05/2021 • 4 minutes to read

Nós

O tipo de hospedagem da Hiperescala (Citus) permite que os servidores do Banco de Dados do Azure para PostgreSQL (chamados de nós) coordenem-se entre si em uma arquitetura "sem compartilhamento". Coletivamente, os nós em um grupo de servidores contêm mais dados e usam mais núcleos de CPU do que seria possível em apenas um servidor. A arquitetura também permite que o banco de dados seja dimensionado adicionando mais nós ao grupo de servidores.

Coordenador e trabalhos

Cada grupo de servidores tem um nó coordenador e vários nós trabalhos. Os aplicativos enviam as consultas ao nó coordenador, que as retransmite para os nós de trabalhos relevantes e acumula os resultados. Os aplicativos não podem se conectar diretamente aos trabalhos.

A Hiperescala (Citus) permite ao administrador de banco de dados *distribuir* tabelas, armazenando linhas diferentes em nós de trabalho diferentes. As tabelas distribuídas são a chave para o desempenho em Hiperescala (Citus). A incapacidade de distribuir tabelas as deixa totalmente no nó de coordenador e sem poder aproveitar o paralelismo entre máquinas.

Para cada consulta em tabelas distribuídas, o coordenador a roteia para um único nó de trabalho ou a coloca em paralelo a vários, depende de onde os dados necessários residem, se em um único nó ou em vários. O coordenador decide o que fazer por meio das tabelas de metadados de consultoria. Essas tabelas acompanham os nomes DNS, a integridade dos nós de trabalho e a distribuição de dados entre nós.

Tipos de tabela

Existem três tipos de tabelas em um grupo de servidores de Hiperescala (Citus). Cada uma armazenada de forma diferente em nós e usada para finalidades diferentes.

Tipo 1: Tabelas distribuídas

O primeiro tipo são tabelas distribuídas e este é o mais comum. Elas parecem ser tabelas normais para instruções SQL, mas são particionados horizontalmente entre nós de trabalho. Isso significa que as linhas da tabela são armazenadas em nós diferentes, em tabelas de fragmento chamadas de fragmentos.

A Hiperescala (Citus) executa instruções SQL e DDL em um cluster. Alterar o esquema de uma tabela distribuída em cascata para atualizar todos os fragmentos da tabela entre os trabalhos.

Coluna de distribuição

A Hiperescala (Citus) usa a fragmentação de algoritmos para atribuir linhas a fragmentos. A atribuição é feita de forma determinista com base no valor de uma coluna de tabela chamada coluna de distribuição. O administrador de cluster deve designar esta coluna ao distribuir uma tabela. É importante fazer a escolha certa por questões de desempenho e funcionalidade.

Tipo 2: tabelas de referência

Uma tabela de referência é um tipo de tabela distribuída cujo conteúdo inteiro está concentrado em um único fragmento. O fragmento é replicado em todos os trabalhos. As consultas em um trabalho podem acessar as informações de referência localmente, sem a sobrecarga de rede de solicitar linhas de outro nó. Tabelas de referência não têm nenhuma coluna de distribuição porque não há necessidade de distinguir fragmentos

separados por linha.

As tabelas de referência em geral são pequenas e usadas para armazenar dados relevantes para consultas em execução em outros nós de trabalho. Um exemplo são valores enumerados, como status de pedidos ou categorias de produtos.

Tipo 3: tabelas locais

Ao usar a Hiperescala (Citus), o nó de coordenador ao qual você se conecta é um Banco de Dados PostgreSQL comum. Você pode criar tabelas comuns no coordenador e optar por não fragmentá-las.

Um bom candidato para tabelas locais seria as tabelas administrativas pequenas que não participam de consultas de junção. Um exemplo é uma tabela de usuários para entrada e autenticação do aplicativo.

Fragments

A seção anterior descreveu como as tabelas distribuídas são armazenadas como fragmentos em nós de trabalho. Esta seção aborda mais detalhes técnicos.

A `pg_dist_shard` tabela de metadados no coordenador contém uma linha para cada fragmento de cada tabela distribuída no sistema. A linha corresponde a uma ID de fragmento com um intervalo de inteiros em um espaço de hash (`shardminvalue`, `shardmaxvalue`).

```
SELECT * from pg_dist_shard;
logicalrelid | shardid | shardstorage | shardminvalue | shardmaxvalue
-----+-----+-----+-----+
github_events | 102026 | t | 268435456 | 402653183
github_events | 102027 | t | 402653184 | 536870911
github_events | 102028 | t | 536870912 | 671088639
github_events | 102029 | t | 671088640 | 805306367
(4 rows)
```

Se o nó de coordenador deseja determinar qual fragmento contém uma linha `github_events`, ele faz o hash do valor da coluna de distribuição na linha. Em seguida, o nó verifica qual intervalo fragmento ' contém o valor do hash. Os intervalos são definidos para que a imagem da função hash seja sua união não contínua.

Posicionamentos de fragmentos

Imagine que o fragmento 102027 esteja associado à linha em questão. A linha é lida ou gravada em uma tabela chamada `github_events_102027` em um dos trabalhos. Em qual trabalho? Isso é totalmente determinado pelas tabelas de metadados. O mapeamento do fragmento para o trabalho é conhecido como posicionamento do fragmento.

O nó coordenador reescreve as consultas em fragmentos que se referem a tabelas específicas como `github_events_102027` e executa esses fragmentos nos trabalhos apropriados. Este é um exemplo de consulta executada nos bastidores para localizar o nó que contém a ID de fragmento 102027.

```
SELECT
    shardid,
    node.nodename,
    node.nodeport
FROM pg_dist_placement placement
JOIN pg_dist_node node
    ON placement.groupid = node.groupid
    AND node.noderole = 'primary'::noderole
WHERE shardid = 102027;
```

shardid	nodename	nodeport
102027	localhost	5433

Próximas etapas

- [Determinar o tipo do aplicativo](#) para preparar a modelagem de dados

Determinando o tipo de aplicativo

09/08/2021 • 2 minutes to read

A execução de consultas eficientes em um grupo de servidores Hiperescala (Citus) exige que as tabelas sejam distribuídas corretamente entre os servidores. A distribuição recomendada varia de acordo com o tipo de aplicativo e seus padrões de consulta.

De modo geral, há dois tipos de aplicativos que funcionam de modo adequado no Hiperescala (Citus). A primeira etapa ao criar uma modelagem de dados é identificar quais deles é mais semelhante ao seu aplicativo.

Visão rápida

APLICATIVOS MULTILOCATÁRIOS	APLICATIVOS EM TEMPO REAL
Às vezes, dezenas ou centenas de tabelas em um esquema	Um número pequeno de tabelas
Consultas relacionadas a um locatário (empresa/repositório) por vez	Consultas de análise relativamente simples com agregações
Cargas de trabalho OLTP para atender clientes Web	Grande volume de ingestão de dados, principalmente os imutáveis
Cargas de trabalho OLAP que atendem consultas analíticas por locatário	Muitas vezes centralizam uma grande tabela de eventos

Exemplos e características

Aplicativos multilocatários

Normalmente, são aplicativos SaaS que atendem a outras empresas, contas ou organizações. A maioria dos aplicativos SaaS são inherentemente relacionais. Eles têm uma dimensão natural para distribuir dados entre nós: apenas fragmento por ID_de locatário.

O Hiperescala (Citus) permite que você escala horizontalmente seu banco de dados para milhões de locatários sem precisar rearquitetar seu aplicativo. Você pode manter a semântica relacional necessária, como junções, restrições de chave estrangeira, transações, ACID e consistência.

- **Exemplos:** sites que hospedam vitrines para outras empresas, como uma solução de marketing digital ou uma ferramenta de automação de vendas.
- **Características:** consultas relacionadas a um locatário em vez de reunir informações de vários locatários. Isso inclui cargas de trabalho OLTP para atender a clientes Web e cargas de trabalho OLAP que atendem a consultas analíticas por locatário. Ter dezenas ou centenas de tabelas em seu esquema de banco de dados também é um indicador para o modelo de dados multilocatário.

Dimensionar um aplicativo multilocatário com o Hiperescala (Citus) também requer alterações mínimas no código do aplicativo. Temos suporte para estruturas populares como Ruby on Rails e Django.

Análise em tempo real

Aplicativos que precisam de um grande paralelismo, coordenando centenas de núcleos para resultados rápidos em consultas numéricas, estatísticas ou de contagem. Fragmentando e paralelizando consultas SQL

em vários nós, o Hiperescala (Citus) possibilita a execução de consultas em tempo real em bilhões de registros em menos de um segundo.

As tabelas em modelos de dados de análise em tempo real normalmente são distribuídas por colunas como _ID, ID_de_locatário ou ID_de_dispositivo.

- **Exemplos:** painéis de análise voltados para o cliente que exigem tempos de resposta de subsegundos.
- **Características:** algumas tabelas, geralmente centralizadas em torno de uma grande tabela de eventos de dispositivo, site ou usuário e que exigem um alto volume de ingestão de dados quase imutáveis. Consultas de análise relativamente simples (mas intensivas do ponto de vista computacional) que envolvem várias agregações e GROUP BYs.

Se a sua situação se assemelhar a ambos os casos acima, a próxima etapa será decidir como fragmentar seus dados no grupo de servidores. A escolha das colunas de distribuição de administrador de banco de dados precisa corresponder aos padrões de acesso de consultas típicas para garantir o desempenho.¹

Próximas etapas

- [Escolha uma coluna de distribuição](#) para tabelas em seu aplicativo para distribuir dados com eficiência

Escolher colunas de distribuição no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 5 minutes to read

Escolher a coluna de distribuição de cada tabela é uma das decisões de modelagem mais importantes que você tomará. O Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) armazena linhas em fragmentos com base no valor da coluna de distribuição das linhas.

A opção correta agrupa os dados relacionados nos mesmos nós físicos, o que torna as consultas rápidas e adiciona suporte a todos os recursos do SQL. Uma opção incorreta faz com que o sistema seja executado lentamente e não dará suporte a todos os recursos do SQL entre os nós.

Este artigo fornece dicas de colunas de distribuição para os dois cenários mais comuns da Hiperescala (Citus).

Aplicativos multilocatário

A arquitetura de multilocatário usa uma forma de modelagem de banco de dados hierárquica para distribuir consultas entre os nós do grupo de servidores. O início da hierarquia de dados é conhecida como a *ID do locatário* e precisa ser armazenada em uma coluna de cada tabela.

A Hiperescala (Citus) inspeciona consultas para ver qual ID de locatário elas envolvem e localiza o fragmento de tabela correspondente. Ela roteia a consulta para um nó de trabalho individual que contém o fragmento. A execução de uma consulta com todos os dados relevantes colocados no mesmo nó é chamada de colocação.

O diagrama a seguir ilustra a colocação no modelo de dados multilocatário. Ele contém duas tabelas, Contas e Campanhas, cada uma distribuída pela `account_id`. As caixas sombreadas representam fragmentos. Os fragmentos verdes são armazenados juntos em um nó de trabalho, e os fragmentos azuis são armazenados em outro nó de trabalho. Observe como uma consulta de junção entre Contas e Campanhas tem todos os dados necessários juntos em um nó quando as duas tabelas são restritas à mesma `account_id`.

Node A

Node B

Accounts table (shard 1)

account_id	name	created_at
1	CNN	2016-07-12
5	Comcast	2016-07-19
...
1252	Walmart	2016-08-02

Accounts table (shard 2)

account_id	name	created_at
2	AT&T	2016-07-13
3	Exxon	2016-07-14
...
1253	UPS	2016-08-03

Campaigns table (shard 3)

campaign_id	name	account_id
1202	tv series	1
1204	superbowl	1
...
352042	chocolate	1252

Campaigns table (shard 4)

campaign_id	name	account_id
2742	gas station	3
2743	my phone	2
...
352423	new phone	2

Para aplicar esse design a um esquema próprio, identifique o que constitui um locatário no seu aplicativo. As instâncias comuns incluem empresa, conta, organização ou cliente. O nome da coluna será semelhante a

`company_id` ou `customer_id`. Examine cada uma das consultas e faça esta pergunta: isso funcionará se ela tiver cláusulas WHERE adicionais para restringir todas as tabelas envolvidas às linhas com a mesma ID de locatário? As consultas no modelo de multilocatário estão no escopo de um locatário. Por exemplo, as consultas sobre as vendas ou o inventário estão no escopo de uma loja específica.

Práticas recomendadas

- **Particione tabelas distribuídas por uma coluna `tenant_id` comum.** Por exemplo, em um aplicativo SaaS em que os locatários são empresas, provavelmente, a `tenant_id` será a `company_id`.
- **Converta pequenas tabelas entre locatários em tabelas de referência.** Quando vários locatários compartilharem uma pequena tabela de informações, distribua-a como uma tabela de referência.
- **Restrinja e filtre todas as consultas de aplicativo por `tenant_id`.** Cada consulta deve solicitar informações para um locatário por vez.

Leia o [tutorial sobre multilocatário](#) para obter um exemplo de como criar esse tipo de aplicativo.

Aplicativos em tempo real

A arquitetura de multilocatário apresenta uma estrutura hierárquica e usa a colocação de dados para rotear consultas por locatário. Por outro lado, as arquiteturas em tempo real dependem de propriedades de distribuição específicas dos dados para obter um processamento altamente paralelo.

Usamos a "ID da entidade" como um termo para colunas de distribuição no modelo em tempo real. As entidades típicas são usuários, hosts ou dispositivos.

As consultas em tempo real normalmente solicitam agregações numéricas agrupadas por data ou categoria. A Hiperescala (Citus) envia essas consultas a cada fragmento para resultados parciais e monta a resposta final no nó coordenador. As consultas são executadas mais rapidamente quando muitos nós contribuem o máximo possível e quando nenhum nó precisa realizar uma quantidade desproporcional de trabalho.

Práticas recomendadas

- **Escolha uma coluna com alta cardinalidade como a coluna de distribuição.** Para comparação, um campo Status em uma tabela de pedidos com os valores Novo, Pago e Enviado é uma opção inadequada de coluna de distribuição. Ele pressupõe apenas esses poucos valores, o que limita o número de fragmentos que podem conter os dados e o número de nós que podem processá-lo. Entre as colunas com alta cardinalidade, também é bom escolher as colunas que são usadas com frequência em cláusulas group-by ou como chaves de junção.
- **Escolha uma coluna com distribuição uniforme.** Se você distribuir uma tabela em uma coluna distorcida para determinados valores comuns, os dados da tabela tenderão a ser acumulados em alguns fragmentos. Os nós que mantêm esses fragmentos acabam realizando mais trabalho do que os outros nós.
- **Distribua tabelas de fatos e dimensões nas colunas comuns.** A tabela de fatos só pode ter uma chave de distribuição. As tabelas unidas em outra chave não serão colocalizadas com a tabela de fatos. Escolha uma dimensão a ser colocalizada com base na frequência na qual ela é unida e no tamanho das linhas de junção.
- **Altere algumas tabelas de dimensões para tabelas de referência.** Se uma tabela de dimensões não puder ser colocalizada com a tabela de fatos, aprimore o desempenho da consulta distribuindo cópias da tabela de dimensões para todos os nós na forma de uma tabela de referência.

Leia o [tutorial sobre o painel em tempo real](#) para obter um exemplo de como criar esse tipo de aplicativo.

Dados de série temporal

Em uma carga de trabalho de série temporal, os aplicativos consultam informações recentes enquanto arquivam informações antigas.

O erro mais comum em modelar informações de série temporal na Hiperescala (Citus) é usar o próprio carimbo de data/hora como uma coluna de distribuição. Uma distribuição de hash com base no tempo distribui os tempos de modo aleatório em fragmentos diferentes, em vez de manter os intervalos de tempo juntos em fragmentos. As consultas que envolvem tempo geralmente referenciam intervalos de tempo, por exemplo, os

dados mais recentes. Esse tipo de distribuição de hash leva à sobrecarga da rede.

Práticas recomendadas

- **Não escolha um carimbo de data/hora como a coluna de distribuição.** Escolha outra coluna de distribuição. Em um aplicativo multilocatário, use a ID do locatário, ou em um aplicativo em tempo real, use a ID da entidade.
- **Em vez disso, use o particionamento de tabela do PostgreSQL para a hora.** Use o particionamento de tabela para dividir uma tabela grande de dados ordenados por hora em várias tabelas herdadas com cada tabela que contém intervalos de tempo diferentes. A distribuição de uma tabela particionada do Postgres na Hiperescala (Citus) cria fragmentos para as tabelas herdadas.

Próximas etapas

- Saiba como a [colocação](#) entre os dados distribuídos ajuda as consultas a serem executadas rapidamente.

Colocação de tabela no Banco de Dados Azure para PostgreSQL - Hiperescala (Citus)

21/05/2021 • 3 minutes to read

Colocação significa armazenar informações relacionadas em conjunto nos mesmos nós. As consultas podem ser rápidas quando todos os dados necessários estão disponíveis sem nenhum tráfego de rede. A colocação de dados relacionados em nós diferentes permite que as consultas executem com eficiência em paralelo em cada nó.

Colocação de dados para tabelas distribuídas por hash

No Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus), uma linha será armazenada em um fragmento se o hash do valor na coluna de distribuição estiver dentro do intervalo de hash do fragmento. Fragmentos com o mesmo intervalo de hash sempre são colocados no mesmo nó. Linhas com valores de coluna de distribuição iguais estão sempre no mesmo nó entre tabelas.

	events shards		page shards	
	shard	hash range	shard	hash range
NODE 1	1	-2147483648 ≤ x ≤ -1073741825	5	-2147483648 ≤ x ≤ -1073741825
	2	-1073741824 ≤ x ≤ -1	6	-1073741824 ≤ x ≤ -1
NODE 2	3	0 ≤ x ≤ 1073741823	7	0 ≤ x ≤ 1073741823
	4	1073741824 ≤ x ≤ 2147483647	8	1073741824 ≤ x ≤ 2147483647

$x = \text{hash}(\text{distribution_column})$

Um exemplo prático de colocação

Considere as tabelas a seguir que podem fazer parte de um SaaS de análise da Web multi-locatário:

```
CREATE TABLE event (
    tenant_id int,
    event_id bigint,
    page_id int,
    payload jsonb,
    primary key (tenant_id, event_id)
);

CREATE TABLE page (
    tenant_id int,
    page_id int,
    path text,
    primary key (tenant_id, page_id)
);
```

Agora, queremos responder a consultas que podem ser emitidas por um painel voltado para o cliente. Um exemplo de consulta é "Retornar o número de visitas na última semana para todas as páginas começando com '/blog' no locatário seis".

Se nossos dados estivesse na opção Single-Server implantação, poderíamos expressar facilmente nossa consulta usando o conjunto rico de operações relacionais oferecido pelo SQL:

```
SELECT page_id, count(event_id)
FROM
  page
LEFT JOIN (
  SELECT * FROM event
  WHERE (payload->>'time')::timestamptz >= now() - interval '1 week'
) recent
USING (tenant_id, page_id)
WHERE tenant_id = 6 AND path LIKE '/blog%'
GROUP BY page_id;
```

Desde que o [conjunto de trabalho](#) para essa consulta caiba na memória, uma tabela de servidor único é uma solução apropriada. Vamos considerar as oportunidades de dimensionar o modelo de dados com a opção Hiperescala (Citus) implantação.

Distribuir tabelas por ID

As consultas de servidor único começam a diminuir conforme o número de locatários e os dados armazenados para cada locatário crescem. O conjunto de trabalho para de se ajustar na memória e a CPU se torna um gargalo.

Nesse caso, podemos fragmento os dados em vários nós usando Hiperescala (Citus). A primeira e mais importante opção que precisamos fazer quando decidirmos o fragmento é a coluna de distribuição. Vamos começar com uma opção simples de usar `event_id` para a tabela de eventos e `page_id` para a `page` tabela:

```
-- naively use event_id and page_id as distribution columns

SELECT create_distributed_table('event', 'event_id');
SELECT create_distributed_table('page', 'page_id');
```

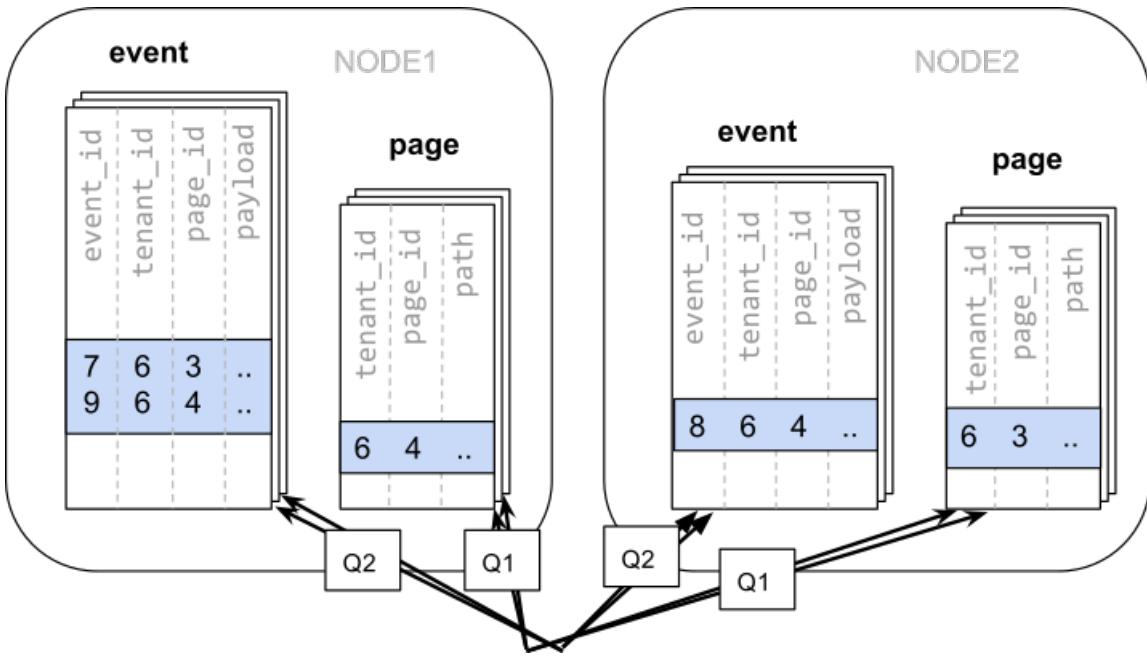
Quando os dados são dispersos em diferentes trabalhadores, não podemos realizar uma junção como faria em um único nó PostgreSQL. Em vez disso, precisamos emitir duas consultas:

```
-- (Q1) get the relevant page_ids
SELECT page_id FROM page WHERE path LIKE '/blog%' AND tenant_id = 6;

-- (Q2) get the counts
SELECT page_id, count(*) AS count
FROM event
WHERE page_id IN /*...page IDs from first query...*/
  AND tenant_id = 6
  AND (payload->>'time')::date >= now() - interval '1 week'
GROUP BY page_id ORDER BY count DESC LIMIT 10;
```

Posteriormente, os resultados das duas etapas precisam ser combinados pelo aplicativo.

Executar as consultas deve consultar dados em fragmentos espalhados entre nós.



Nesse caso, a distribuição de dados cria desvantagens substanciais:

- Sobrecarga da consulta de cada fragmento e execução de várias consultas.
- Sobrecarga de Q1 retornando muitas linhas para o cliente.
- Q2 torna-se grande.
- A necessidade de escrever consultas em várias etapas requer alterações no aplicativo.

Os dados são dispersos, portanto, as consultas podem ser paralelizadas. Só é benéfico se a quantidade de trabalho que a consulta faz é consideravelmente maior do que a sobrecarga de consultar muitos fragmentos.

Distribuir tabelas por locatário

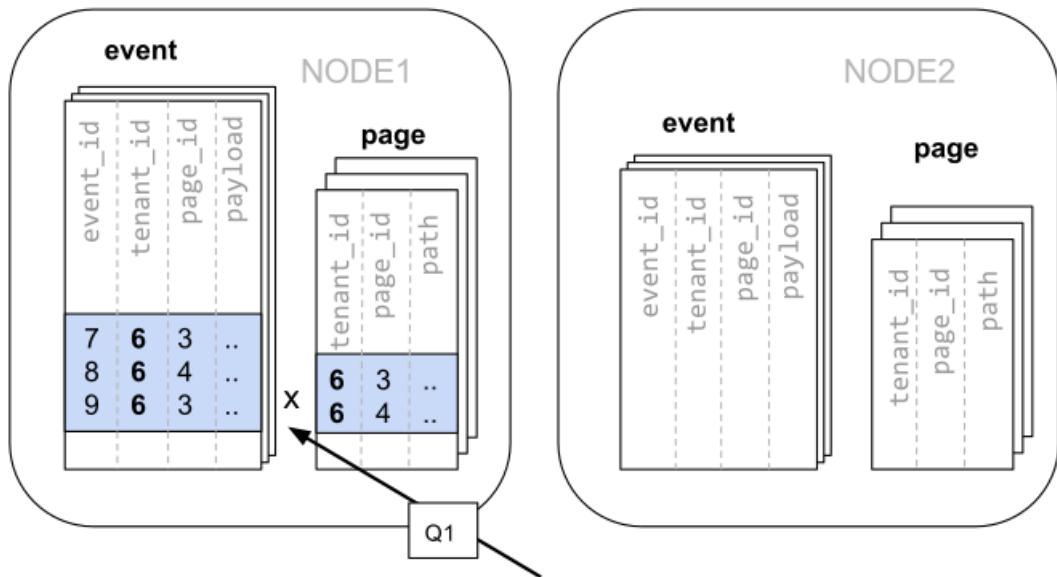
Em hiperescala (Citus), as linhas com o mesmo valor de coluna de distribuição têm a garantia de estar no mesmo nó. A partir de então, podemos criar nossas tabelas com `tenant_id` como a coluna de distribuição.

```
-- co-locate tables by using a common distribution column
SELECT create_distributed_table('event', 'tenant_id');
SELECT create_distributed_table('page', 'tenant_id', colocate_with => 'event');
```

Agora o Citus (subscale) pode responder à consulta de servidor único original sem modificação (Q1):

```
SELECT page_id, count(event_id)
FROM
  page
LEFT JOIN (
  SELECT * FROM event
  WHERE (payload->>'time')::timestamptz >= now() - interval '1 week'
) recent
USING (tenant_id, page_id)
WHERE tenant_id = 6 AND path LIKE '/blog%'
GROUP BY page_id;
```

Devido ao filtro e à junção em `tenant_id`, o Citus (hiperescala) sabe que toda a consulta pode ser respondida usando o conjunto de fragmentos colocais que contêm os dados para esse locatário específico. Um único nó PostgreSQL pode responder à consulta em uma única etapa.



Em alguns casos, consultas e esquemas de tabela devem ser alterados para incluir a ID do locatário em restrições exclusivas e condições de junção. Essa alteração geralmente é simples.

Próximas etapas

- Veja como os dados de locatário são colocalizados no [tutorial multi-locatário](#).

Regras de firewall no Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)

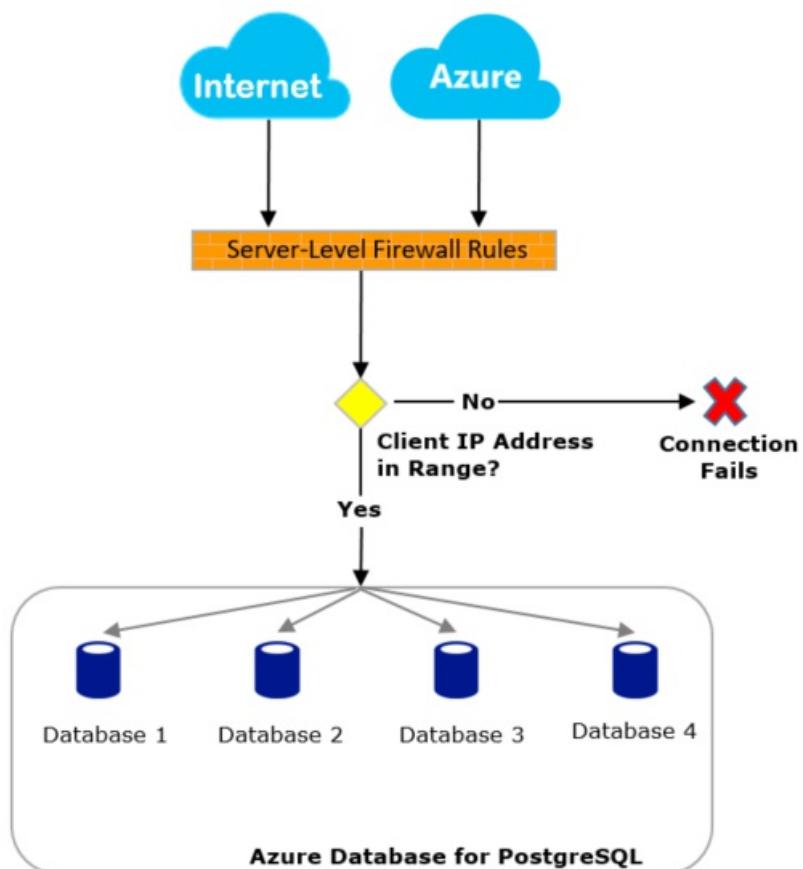
21/05/2021 • 3 minutes to read

O firewall do servidor do Banco de Dados do Azure para PostgreSQL impede todos os acessos ao seu nó coordenador de Hiperescala (Citus) até você especificar quais computadores têm permissão. O firewall concede acesso ao servidor com base no endereço IP de origem de cada solicitação. Para configurar seu firewall, você deve criar regras de firewall que especifiquem intervalos de endereços IP aceitáveis. Você pode criar regras de firewall no nível de servidor.

Regras de firewall: essas regras permitem que os clientes acessem o nó coordenador de Hiperescala (Citus), ou seja, todos os bancos dentro do mesmo servidor lógico. As regras de firewall no nível de servidor podem ser configuradas por meio do portal do Azure. Para criar regras de firewall no nível de servidor, você deve ser o proprietário da assinatura ou um colaborador da assinatura.

Visão geral do firewall

Todo acesso de banco de dados ao seu nó coordenador é bloqueado por padrão pelo firewall. Para começar a usar o servidor de outro computador, especifique uma ou mais regras de firewall no nível do servidor para permitir o acesso ao seu servidor. Use as regras de firewall para especificar quais intervalos de endereços IP da Internet permitir. O acesso em si ao site do Portal do Azure não é afetado pelas regras de firewall. As tentativas de conexão da Internet e do Azure devem passar primeiramente pelo firewall antes de poderem acessar seu Banco de Dados PostgreSQL, conforme exibido no diagrama a seguir:



Conectar pela Internet e pelo Azure

Um firewall do grupo de servidores de Hiperescala (Citus) controla quem pode se conectar ao nó coordenador do grupo. O firewall determina o acesso consultando uma lista configurável de regras. Cada regra é um endereço IP ou um intervalo de endereços que são permitidos.

Quando o firewall bloqueia conexões, isso pode causar erros de aplicativo. O uso do driver JDBC do PostgreSQL, por exemplo, gera um erro como este:

```
java.util.concurrent.ExecutionException: java.lang.RuntimeException: org.postgresql.util.PSQLException:  
FATAL: no pg_hba.conf entry for host "123.45.67.890", user "citus", database "citus", SSL
```

Consulte [Criar e gerenciar regras de firewall](#) para saber como as regras são definidas.

Solução de problemas do firewall do servidor de banco de dados

Quando o acesso ao serviço do Banco de Dados do Microsoft Azure para PostgreSQL - Hiperescala (Citus) não se comporta conforme o esperado, considere estes pontos:

- **As alterações na lista de permissões ainda não entraram em vigor:** pode ocorrer um atraso máximo de cinco minutos para que as alterações na configuração de Hiperescala (Citus) entrem em vigor.
- **O usuário não está autorizado ou uma senha incorreta foi usada:** se um usuário não tiver permissões no servidor ou se a senha usada estiver incorreta, a conexão ao servidor será negada. Criar uma configuração de firewall apenas oferece aos clientes uma oportunidade de tentar se conectar ao servidor; cada cliente ainda deverá fornecer as credenciais de segurança necessárias.

Por exemplo, usando um cliente JDBC, o seguinte erro pode aparecer.

```
java.util.concurrent.ExecutionException: java.lang.RuntimeException: org.postgresql.util.PSQLException:  
FATAL: falha na autenticação da senha para o usuário "seunomedesusário"
```

- **Endereço IP dinâmico:** se você tiver uma conexão com a Internet com endereçamento IP dinâmico e estiver com dificuldades para atravessar o firewall, tente uma das seguintes soluções:
 - Solicite ao provedor de serviços de Internet (ISP) o intervalo de endereços IP atribuído aos computadores clientes que acessarão o nó coordenador de Hiperescala (Citus) e depois adicione o intervalo de endereços IP como uma regra de firewall.
 - Obtenha o endereçamento IP estático para os computadores cliente e adicione os endereços IP estáticos como uma regra de firewall.

Próximas etapas

Para ver artigos sobre como criar regras de firewall no nível de servidor e de banco de dados, consulte:

- [Criar e gerenciar regras de firewall do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure](#)

Configurar o TLS no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 2 minutes to read

O nó coordenador da Hiperescala (Citus) exige que os aplicativos clientes se conectem com o protocolo TLS (Segurança da Camada de Transporte). O uso do protocolo TLS entre o servidor de banco de dados e os aplicativos clientes ajuda a manter o sigilo dos dados em trânsito. As configurações de verificação adicionais descritas abaixo também protegem contra ataques "man-in-the-middle".

Impondo conexões do protocolo TLS

Os aplicativos usam uma "cadeia de conexão" para identificar o banco de dados de destino e as configurações da conexão. Clientes diferentes exigem configurações diferentes. Veja a lista de cadeias de conexão usadas pelos clientes mais comuns na seção **Cadeias de conexão** do seu grupo de servidores no portal do Azure.

Os parâmetros do TLS `ssl` e `sslmode` variam de acordo com os recursos do conector, por exemplo `ssl=true`, `sslmode=require` ou `sslmode=required`.

Verificar se o seu aplicativo ou sua estrutura oferece suporte a conexões TLS

Algumas estruturas de aplicativo não habilitam o TLS por padrão em conexões do PostgreSQL. No entanto, sem uma conexão segura, o aplicativo não pode se conectar a um nó coordenador da Hiperescala (Citus). Confira a documentação de seu aplicativo para saber como habilitar conexões TLS.

Aplicativos que exigem a verificação de certificado para conectividade TLS

Em alguns casos, os aplicativos exigem um arquivo de certificado local gerado de um arquivo de certificado (.cer) de uma Autoridade de Certificação (CA) confiável para se conectar com segurança. O certificado para se conectar a um servidor de Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) está localizado em <https://cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem>. Baixe o arquivo de certificado e salve-o em seu local preferido.

NOTE

Para verificar a autenticidade do certificado, use a ferramenta de linha de comando OpenSSL para verificar a impressão digital SHA-256:

```
openssl x509 -in DigiCertGlobalRootCA.crt.pem -noout -sha256 -fingerprint  
# should output:  
# 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:61
```

Conecte-se usando psql

O exemplo abaixo mostra como se conectar ao nó coordenador da Hiperescala (Citus) com o utilitário de linha de comando psql. Use a configuração de cadeia de conexão `sslmode=verify-full` para impor a verificação de certificado TLS. Passe o caminho do arquivo de certificado local para o parâmetro `sslrootcert`.

Veja abaixo um exemplo da cadeia de conexão do psql:

```
psql "sslmode=verify-full sslrootcert=DigiCertGlobalRootCA.crt.pem  
host=mydemoserver.postgres.database.azure.com dbname=citus user=citus password=your_pass"
```

TIP

Confirme se o valor passado para `sslrootcert` corresponde ao caminho do arquivo para o certificado que você salvou.

Próximas etapas

Aumente ainda mais a segurança com [as regras de firewall Banco de Dados do Azure para PostgreSQL – Hiperescala \(Citus\)](#).

Manutenção agendada no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

09/08/2021 • 2 minutes to read

O Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) faz manutenção periódica para manter seu banco de dados gerenciado seguro, estável e atualizado. Durante a manutenção, todos os nós no grupo de servidores obtêm novos recursos, atualizações e patches.

Os principais recursos da manutenção agendada para a Hiperescala (Citus) são:

- Atualizações são aplicadas ao mesmo tempo em todos os nós no grupo de servidores
- Notificações de manutenção futura são postadas na Integridade do Serviço do Azure com cinco dias de antecedência
- Normalmente, há pelo menos 30 dias entre eventos de manutenção bem-sucedidos para um grupo de servidores
- O dia preferencial da semana e o intervalo de tempo nesse dia para o início da manutenção podem ser definidos para cada grupo de servidores individualmente

Selecionar uma janela de manutenção e uma notificação sobre a próxima manutenção

Você pode agendar uma manutenção durante um dia específico da semana e um período dentro desse dia. Ou pode permitir que o sistema escolha um dia e um período para você automaticamente. Seja como for, o sistema alertará você cinco dias antes de executar qualquer manutenção. O sistema também avisará quando a manutenção for iniciada e quando for concluída com êxito.

Notificações de manutenção agendada futura são postadas na Integridade do Serviço do Azure e podem ser:

- Enviadas por email para um endereço específico
- Enviadas por email para uma função do Azure Resource Manager
- Enviadas em um SMS (mensagem de texto) para dispositivos móveis
- Envio por push como notificação para um aplicativo do Azure
- Entregue como mensagem de voz

Ao especificar preferências para o agendamento de manutenção, você pode escolher um dia da semana e uma janela de tempo. Se você não especificar, o sistema escolherá os horários entre 23h e 7h no horário da região do grupo do servidor. Defina diferentes cronogramas para cada grupo de servidores de Hiperescala (Citus) na assinatura do Azure.

IMPORTANT

Normalmente, há pelo menos 30 dias entre os eventos de manutenção agendada bem-sucedidos para um grupo de servidores.

No entanto, no caso de uma atualização de emergência crítica, como uma vulnerabilidade grave, a janela de notificação pode ter menos de cinco dias. A atualização crítica pode ser aplicada ao seu servidor, mesmo se uma manutenção agendada bem-sucedida tiver sido realizada nos últimos 30 dias.

Você pode atualizar as configurações de agendamento a qualquer momento. Caso haja uma manutenção

agendada para seu grupo de servidores de Hiperescala (Citus) e você atualizar o agendamento, os eventos existentes continuarão conforme o agendamento original. A alteração das configurações entrará em vigor após a conclusão dos eventos existentes.

Se a manutenção falhar ou for cancelada, o sistema criará uma notificação. Ele tentará realizar a manutenção novamente de acordo com as configurações de agendamento atuais e notificará o próximo evento de manutenção com cinco dias de antecedência.

Próximas etapas

- Saiba como [alterar o agendamento de manutenção](#)
- Saiba como [receber notificações sobre manutenções futuras](#) usando a Integridade do Serviço do Azure
- Saiba como [configurar alertas sobre eventos futuros de manutenção agendada](#)

Backup e restauração no Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)

12/08/2021 • 2 minutes to read

O Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) cria automaticamente backups de cada nó e os coloca em um armazenamento com redundância local. Os backups podem ser usados para restaurar seu cluster de Hiperescala (Citus) para um horário especificado. Os recursos de backup e restauração são uma parte essencial de qualquer estratégia de continuidade dos negócios, pois eles protegem seus dados contra exclusão ou corrupção acidentais.

Backups

Pelo menos uma vez por dia, o Banco de Dados do Azure para PostgreSQL faz instantâneos de backup dos arquivos de dados e do log de transações do banco de dados. Os backups permitem restaurar um servidor pontualmente dentro do período de retenção. (No momento, o período de retenção é de 35 dias para todos os grupos de servidores.) Todos os backups são criptografados usando a criptografia AES de 256 bits.

Nas regiões do Azure que dão suporte a zonas de disponibilidade, os instantâneos de backup são armazenados em três zonas de disponibilidade. Se pelo menos uma zona de disponibilidade estiver online, o cluster de Hiperescala (Citus) será restaurável.

Os arquivos de backup não podem ser exportados. Eles só podem ser usados para operações de restauração no Banco de Dados do Azure para PostgreSQL.

Custo do armazenamento de backup

Para obter o preço atual do armazenamento de backup, consulte a [página de preços](#) do Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus).

Restaurar

Você pode restaurar um grupo de servidores de Hiperescala (Citus) para qualquer ponto no tempo nos últimos 35 dias. A Restauração pontual é útil em vários cenários. Por exemplo, quando um usuário exclui dados, remove uma tabela ou um banco de dados importante por engano ou se um aplicativo substitui, acidentalmente, dados válidos por dados inválidos.

IMPORTANT

Os grupos de servidores de Hiperescala (Citus) excluídos não podem ser restaurados. Ao excluir o grupo de servidores, todos os nós que pertencem a ele são excluídos e não é possível recuperá-los. Para proteger os recursos do servidor, após a implantação, da exclusão acidental ou de alterações inesperadas, os administradores podem usar os [bloqueios de gerenciamento](#).

O processo de restauração cria um grupo de servidores na mesma região, assinatura e grupo de recursos do Azure que o original. O grupo de servidores tem a configuração original: o mesmo número de nós, número de vCores, tamanho de armazenamento, funções de usuário, versão do PostgreSQL e versão da extensão Citus.

As configurações de firewall e os parâmetros do servidor PostgreSQL não são preservados do grupo de servidores original e são redefinidos para valores padrão. O firewall impedirá todas as conexões. Você precisará ajustar manualmente essas configurações após a restauração. Em geral, confira nossa lista de [tarefas de pós-restauração](#) sugeridas.

Próximas etapas

- Confira as etapas para [restaurar um grupo de servidores](#) no portal do Azure.
- Saiba mais sobre as [Zonas de disponibilidade do Azure](#).

Alta disponibilidade em Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 2 minutes to read

A HA (alta disponibilidade) evita o tempo de inatividade no banco de dados mantendo réplicas em espera de cada nó em um grupo de servidores. Se um nó falhar, o Hiperescala (Citus) alterna as conexões de entrada do nó com falha para o modo em espera. O failover ocorre em alguns minutos e os nós promovidos sempre têm dados atualizados por meio de streaming de replicação síncrona do PostgreSQL.

Afim de aproveitar a HA no nó de coordenador, os aplicativos de banco de dados precisam detectar e repetir conexões descartadas e transações com falha. Será possível acessar o coordenador recém-promovido com a mesma cadeia de conexão.

A recuperação pode ser dividida em três estágios: detecção, failover e recuperação completa. A Hiperescala (Citus) executa verificações de integridade periódicas em cada nó e após quatro verificações com falha ele determina que um nó está inoperante. Em seguida, a Hiperescala (Citus) promove um modo de espera para o status do nó primário (failover) e provisiona uma nova espera. A replicação de streaming começa ao colocar o novo nó atualizado. Quando todos os dados forem replicados, o nó atingirá a recuperação completa.

Próximas etapas

- Saiba mais sobre como [habilitar a alta disponibilidade](#) em um grupo de servidores de Hiperescala (Citus).

Réplicas de leitura no Banco de Dados PostgreSQL do Azure - Hiperescala (Citus)

21/05/2021 • 4 minutes to read

IMPORTANT

As réplicas de leitura no Hiperescala (Citus) estão atualmente em versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

Você pode ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

O recurso de réplica de leitura permite replicar dados de um servidor do Hiperescala (Citus) para um servidor somente leitura. As réplicas são atualizadas de forma assíncrona com a tecnologia de replicação física do PostgreSQL. Você pode replicar do servidor primário para um número ilimitado de réplicas.

Réplicas são novos grupos de servidores que você gerencia de forma semelhante a grupos de servidores Hiperescala (Citus) regulares. Para cada réplica de leitura, você será cobrado pela computação provisionada em vCores e pelo armazenamento em GB/mês.

Saiba como [criar e gerenciar réplicas](#).

Quando usar uma réplica de leitura

O recurso de réplica de leitura ajuda a melhorar o desempenho e o dimensionamento de cargas de trabalho com uso intenso de leitura. As cargas de trabalho de leitura podem ser isoladas para as réplicas, enquanto as cargas de trabalho de gravação podem ser direcionadas para o primário.

Um cenário comum é ter cargas de trabalho analíticas e de BI usando a réplica de leitura como a fonte de dados para relatório.

Como réplicas são somente leitura, elas não reduzem diretamente os encargos de capacidade de gravação no primário.

Considerações

O recurso destina-se a cenários em que o atraso da replicação é aceitável e para descarregamento de consultas. Não é destinado a cenários de replicação síncrona em que os dados de réplica devem estar atualizados. Haverá um atraso mensurável entre o primário e a réplica. Esse atraso pode ser em minutos ou até mesmo em horas, dependendo da carga de trabalho e da latência entre o primário e a réplica. Os dados na réplica acabarão se tornando consistentes com os dados no primário. Use este recurso para cargas de trabalho que podem acomodar esse atraso.

Criar uma réplica

Quando você inicia o fluxo de trabalho de criação de réplica, um grupo de servidores Hiperescala (Citus) é criado. O novo grupo é preenchido com os dados que estavam no grupo de servidores primário. A hora de criação depende da quantidade de dados no primário e do tempo decorrido desde o último backup completo semanal. O tempo pode variar de alguns minutos a várias horas.

O recurso de réplica de leitura usa a replicação física do PostgreSQL, e não a replicação lógica. O modo padrão é replicação de streaming usando slots de replicação. Quando necessário, o envio de logs é usado para recuperar

o atraso.

Saiba como [criar uma réplica de leitura no portal do Azure](#).

Conectar-se a uma réplica

Quando você cria uma réplica, ela não herda as regras de firewall do grupo de servidores primário. Essas regras precisam ser configuradas independentemente da réplica.

A réplica herda a conta do administrador ("Citus") do grupo de servidores primário. Todas as contas de usuário são replicadas para as réplicas de leitura. Você só pode se conectar a uma réplica de leitura usando as contas de usuário disponíveis no servidor primário.

Você pode se conectar ao nó coordenador da réplica usando seu nome de host e uma conta de usuário válida, como faria em um grupo de servidores Hiperescala (Citus). Para um servidor chamado **my replica** com o nome de usuário administrador **citus**, você pode se conectar ao nó coordenador da réplica usando o psql:

```
psql -h c.myreplica.postgres.database.azure.com -U citus@myreplica -d postgres
```

No prompt, insira a senha da conta de usuário.

Considerações

Esta seção resume as considerações sobre o recurso de réplica de leitura.

Novas réplicas

Uma réplica de leitura é criada como um novo grupo de servidores Hiperescala (Citus). Um grupo de servidores existente não pode se tornar uma réplica. Você não pode criar uma réplica de outra réplica de leitura.

Configuração da réplica

Uma réplica é criada usando as mesmas configurações de computação, armazenamento e nó de trabalho que o primário. Depois que uma réplica é criada, várias configurações podem ser alteradas, incluindo o período de retenção do armazenamento e do backup. Outras configurações não podem ser alteradas nas réplicas, como o tamanho do armazenamento e o número de nós de trabalho.

Lembre-se de manter as réplicas suficientemente robustas para manter as alterações que chegam do primário. Por exemplo, lembre-se de aumentar o poder de computação nas réplicas se ele for aumentado no primário.

Regras de firewall e configurações de parâmetros não são herdadas do servidor primário para a réplica quando a réplica é criada ou posteriormente.

Regiões

Os grupos de servidores Hiperescala (Citus) permitem apenas a replicação da mesma região.

Próximas etapas

- Saiba como [criar e gerenciar réplicas de leitura no portal do Azure](#).

Monitoramento e ajuste do Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 2 minutes to read

Monitorar os dados dos seus servidores ajuda a solucionar problemas e otimizar sua carga de trabalho. O Hiperescala (Citus) fornece várias opções de monitoramento para fornecer insights sobre o comportamento de nós em um grupo de servidores.

Métricas

O Hiperescala (Citus) fornece métricas para cada nó em um grupo de servidores. As métricas fornecem informações sobre o comportamento dos recursos de suporte. Cada métrica é emitida em uma frequência de um minuto e tem até 30 dias de histórico.

Além de exibir grafos das métricas, você pode configurar alertas. Para obter diretrizes passo a passo, consulte [How to set up alerts](#) (Como configurar alertas). Outras tarefas incluem a configuração de ações automatizadas, execução de análises avançadas e arquivamento de histórico. Para obter mais informações, consulte a [Visão geral das métricas no Microsoft Azure](#).

Listá de métricas

Essas métricas estão disponíveis para nós de Hiperescala (Citus):

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRÍÇÃO
active_connections	Conexões ativas	Contagem	O número de conexões ativas com o servidor.
cpu_percent	Porcentagem de CPU	Porcentagem	O percentual de CPU em uso.
iops	IOPS	Contagem	Consulte a definição de IOPS e a taxa de transferência do Hiperescala (Citus)
memory_percent	Porcentagem de memória	Porcentagem	O percentual de memória em uso.
network_bytes_ingress	Entrada na rede	Bytes	Entrada de rede em conexões ativas.
network_bytes_egress	Saída da rede	Bytes	Rede-Out em conexões ativas.
storage_percent	Porcentagem de armazenamento	Porcentagem	O percentual de armazenamento usado fora do máximo do servidor.

MÉTRICA	NOME DE EXIBIÇÃO DA MÉTRICA	UNIDADE	DESCRIÇÃO
storage_used	Armazenamento usado	Bytes	A quantidade de armazenamento em uso. O armazenamento usado pelo serviço pode incluir os arquivos de banco de dados, os logs de transação e os logs do servidor.

O Azure não fornece métricas agregadas para o cluster como um todo, mas as métricas para vários nós podem ser colocadas no mesmo grafo.

Próximas etapas

- Confira [como configurar alertas](#) para obter orientação sobre como criar um alerta em uma métrica.

Registro de auditoria no Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)

09/08/2021 • 3 minutes to read

Os logs de auditoria de atividades do banco de dados no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) estão disponíveis por meio da extensão de Auditoria do PostgreSQL: [pgAudit](#). O pgAudit fornece o log de auditoria detalhado de sessões e/ou objetos.

IMPORTANT

O pgAudit está em versão preliminar no Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)

Se você quiser logs no nível de recursos do Azure para operações como o dimensionamento de computação e armazenamento, veja o [Log de atividades do Azure](#).

Considerações sobre o uso

Por padrão, as instruções de log pgAudit são emitidas junto com suas instruções de log regulares usando o recurso de log padrão do Postgres. No Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus), você pode configurar todos os logs a serem enviados para o repositório de logs do Azure Monitor para análise posterior no Log Analytics. Se você habilitar Azure Monitor log de recursos, seus logs serão enviados automaticamente (no formato JSON) para o Armazenamento do Microsoft Azure, Hubs de Eventos e/ou logs de Azure Monitor, dependendo de sua escolha.

Habilitar o pgAudit

A extensão pgAudit é pré-instalada e habilitada em todos os nós do grupo de servidores do Hiperescala (Citus). Nenhuma ação é necessária para habilitá-la.

Configurações do pgAudit

O pgAudit permite que você configure o log de auditoria de sessão ou objeto. O [log de auditoria de sessão](#) emite logs detalhados de instruções executadas. O [log de auditoria de objeto](#) tem o escopo de auditoria para relações específicas. Você pode optar por configurar um ou ambos os tipos de registro em log.

NOTE

As configurações do pgAudit são especificadas globalmente e não podem ser especificadas em um nível de banco de dados ou de função.

Além disso, as configurações da pgAudit são especificadas por nó em um grupo de servidores. Para fazer uma alteração em todos os nós, você deve aplicá-la a cada nó individualmente.

Você deve configurar os parâmetros de pgAudit para iniciar o registro em log. A [documentação do pgAudit](#) fornece a definição de cada parâmetro. Teste os parâmetros primeiro e confirme que você está obtendo o comportamento esperado.

NOTE

Definir `pgaudit.log_client` como ATIVADO redirecionará os logs para um processo de cliente, como `psql`, em vez de serem gravados no arquivo. Essa configuração deve ser deixada desabilitada.

`pgaudit.log_level` é habilitado somente quando `pgaudit.log_client` está ativado.

NOTE

No Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus), `pgaudit.log` não pode ser definido usando um atalho de sinal (menos) `-`, conforme descrito na documentação do pgAudit. Todas as classes de instrução necessárias (LEITURA, GRAVAÇÃO etc.) devem ser especificadas individualmente.

Formato do log de auditoria

Cada entrada de auditoria é indicado `AUDIT:` próximo ao início da linha de log. O formato do restante da entrada é detalhado na [documentação do pgAudit](#).

Introdução

Para começar rapidamente, defina `pgaudit.log` como `WRITE` e abra seus logs de servidor para examinar a saída.

Exibir logs de auditoria

A maneira como você acessa os logs depende do ponto de extremidade escolhido. Para o Armazenamento do Microsoft Azure, veja o artigo sobre [conta de armazenamento de logs](#). Para os hubs de eventos, veja o artigo sobre [fluxos de logs do Azure](#).

Para logs de Azure Monitor, os logs são enviados para o espaço de trabalho selecionado. Os logs do Postgres usam o modo de coleta `AzureDiagnostics` para que possam ser consultados a partir da tabela `AzureDiagnostics`. Os campos na tabela são descritos abaixo. Saiba mais sobre como consultar e alertar na visão geral [Consulta de logs do Azure Monitor](#).

Você pode usar essa consulta para começar. Você pode configurar alertas com base em consultas.

Pesquisar em todas as entradas do pgAudit nos logs do Postgres por um servidor específico no último dia

```
AzureDiagnostics
| where LogicalServerName_s == "myservername"
| where TimeGenerated > ago(1d)
| where Message contains "AUDIT:"
```

Próximas etapas

- [Saiba como configurar o registro em log no Banco de Dados do Azure para PostgreSQL – Hiperescala \(Citus\)](#) e como acessar os logs

Armazenamento de tabelas colunar (versão prévia)

09/08/2021 • 8 minutes to read

IMPORTANT

O Armazenamento de tabelas colunar em Hiperescala (Citus) está na versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

Você pode ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

O Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus) dá suporte apenas ao armazenamento de tabelas colunar com acréscimo para cargas de trabalho analíticas e de data warehouse. Quando colunas (em vez de linhas) são armazenadas contiguamente no disco, os dados ficam mais compactáveis e as consultas podem solicitar um subconjunto de colunas mais rapidamente.

Uso

Para usar o armazenamento colunar, especifique `USING columnar` ao criar uma tabela:

```
CREATE TABLE contestant (
    handle TEXT,
    birthdate DATE,
    rating INT,
    percentile FLOAT,
    country CHAR(3),
    achievements TEXT[]
) USING columnar;
```

A Hiperescala (Citus) converte linhas em armazenamento colunar em "faixas" durante a inserção. Cada faixa mantém dados equivalentes aos de uma transação ou 150.000 linhas, o que for menor. (O tamanho da faixa e outros parâmetros de uma tabela colunar podem ser alterados com a função [alter_columnar_table_set](#).)

Por exemplo, a instrução abaixo coloca todas as cinco linhas na mesma faixa porque todos os valores estão inseridos em uma única transação:

```
-- insert these values into a single columnar stripe

INSERT INTO contestant VALUES
('a', '1990-01-10', 2090, 97.1, 'XA', '{a}'),
('b', '1990-11-01', 2203, 98.1, 'XA', '{a,b}'),
('c', '1988-11-01', 2907, 99.4, 'XB', '{w,y}'),
('d', '1985-05-05', 2314, 98.3, 'XB', '{}'),
('e', '1995-05-05', 2236, 98.2, 'XC', '{a}'');
```

É melhor criar faixas mais largas sempre que possível, pois a Hiperescala (Citus) comprime dados colunares separadamente por cada faixa. Podemos ver fatos sobre nossa tabela colunar, como taxa de compactação, número de faixas e média de linhas por faixa, usando `VACUUM VERBOSE`:

```
VACUUM VERBOSE contestant;
```

```
INFO: statistics for "contestant":  
storage id: 10000000000  
total file size: 24576, total data size: 248  
compression rate: 1.31x  
total row count: 5, stripe count: 1, average rows per stripe: 5  
chunk count: 6, containing data for dropped columns: 0, zstd compressed: 6
```

A saída mostra que a Hiperescala (Citus) usou o algoritmo de compactação zstd para alcançar a compactação de dados de 1,31x. A taxa de compactação compara a) o tamanho dos dados inseridos como estavam preparados na memória com b) o tamanho dos dados compactados na sua faixa.

Considerando como é medida, a taxa de compactação pode ou não corresponder à diferença de tamanho entre o armazenamento de linha e colunar de uma tabela. A única maneira de realmente descobrir essa diferença é criar uma tabela colunar e de linha que contenha os mesmos dados e comparar.

Medir a compactação

Vamos criar um exemplo com mais dados para fazer o parâmetro de comparação da economia de compactação.

```
-- first a wide table using row storage  
CREATE TABLE perf_row(  
    c00 int8, c01 int8, c02 int8, c03 int8, c04 int8, c05 int8, c06 int8, c07 int8, c08 int8, c09 int8,  
    c10 int8, c11 int8, c12 int8, c13 int8, c14 int8, c15 int8, c16 int8, c17 int8, c18 int8, c19 int8,  
    c20 int8, c21 int8, c22 int8, c23 int8, c24 int8, c25 int8, c26 int8, c27 int8, c28 int8, c29 int8,  
    c30 int8, c31 int8, c32 int8, c33 int8, c34 int8, c35 int8, c36 int8, c37 int8, c38 int8, c39 int8,  
    c40 int8, c41 int8, c42 int8, c43 int8, c44 int8, c45 int8, c46 int8, c47 int8, c48 int8, c49 int8,  
    c50 int8, c51 int8, c52 int8, c53 int8, c54 int8, c55 int8, c56 int8, c57 int8, c58 int8, c59 int8,  
    c60 int8, c61 int8, c62 int8, c63 int8, c64 int8, c65 int8, c66 int8, c67 int8, c68 int8, c69 int8,  
    c70 int8, c71 int8, c72 int8, c73 int8, c74 int8, c75 int8, c76 int8, c77 int8, c78 int8, c79 int8,  
    c80 int8, c81 int8, c82 int8, c83 int8, c84 int8, c85 int8, c86 int8, c87 int8, c88 int8, c89 int8,  
    c90 int8, c91 int8, c92 int8, c93 int8, c94 int8, c95 int8, c96 int8, c97 int8, c98 int8, c99 int8  
);  
  
-- next a table with identical columns using columnar storage  
CREATE TABLE perf_columnar(LIKE perf_row) USING COLUMNAR;
```

Preencha ambas as tabelas com o mesmo grande conjuntos de dados:

```

INSERT INTO perf_row
SELECT
    g % 00500, g % 01000, g % 01500, g % 02000, g % 02500, g % 03000, g % 03500, g % 04000, g % 04500, g %
05000,
    g % 05500, g % 06000, g % 06500, g % 07000, g % 07500, g % 08000, g % 08500, g % 09000, g % 09500, g %
10000,
    g % 10500, g % 11000, g % 11500, g % 12000, g % 12500, g % 13000, g % 13500, g % 14000, g % 14500, g %
15000,
    g % 15500, g % 16000, g % 16500, g % 17000, g % 17500, g % 18000, g % 18500, g % 19000, g % 19500, g %
20000,
    g % 20500, g % 21000, g % 21500, g % 22000, g % 22500, g % 23000, g % 23500, g % 24000, g % 24500, g %
25000,
    g % 25500, g % 26000, g % 26500, g % 27000, g % 27500, g % 28000, g % 28500, g % 29000, g % 29500, g %
30000,
    g % 30500, g % 31000, g % 31500, g % 32000, g % 32500, g % 33000, g % 33500, g % 34000, g % 34500, g %
35000,
    g % 35500, g % 36000, g % 36500, g % 37000, g % 37500, g % 38000, g % 38500, g % 39000, g % 39500, g %
40000,
    g % 40500, g % 41000, g % 41500, g % 42000, g % 42500, g % 43000, g % 43500, g % 44000, g % 44500, g %
45000,
    g % 45500, g % 46000, g % 46500, g % 47000, g % 47500, g % 48000, g % 48500, g % 49000, g % 49500, g %
50000
FROM generate_series(1,50000000) g;

INSERT INTO perf_columnar
SELECT
    g % 00500, g % 01000, g % 01500, g % 02000, g % 02500, g % 03000, g % 03500, g % 04000, g % 04500, g %
05000,
    g % 05500, g % 06000, g % 06500, g % 07000, g % 07500, g % 08000, g % 08500, g % 09000, g % 09500, g %
10000,
    g % 10500, g % 11000, g % 11500, g % 12000, g % 12500, g % 13000, g % 13500, g % 14000, g % 14500, g %
15000,
    g % 15500, g % 16000, g % 16500, g % 17000, g % 17500, g % 18000, g % 18500, g % 19000, g % 19500, g %
20000,
    g % 20500, g % 21000, g % 21500, g % 22000, g % 22500, g % 23000, g % 23500, g % 24000, g % 24500, g %
25000,
    g % 25500, g % 26000, g % 26500, g % 27000, g % 27500, g % 28000, g % 28500, g % 29000, g % 29500, g %
30000,
    g % 30500, g % 31000, g % 31500, g % 32000, g % 32500, g % 33000, g % 33500, g % 34000, g % 34500, g %
35000,
    g % 35500, g % 36000, g % 36500, g % 37000, g % 37500, g % 38000, g % 38500, g % 39000, g % 39500, g %
40000,
    g % 40500, g % 41000, g % 41500, g % 42000, g % 42500, g % 43000, g % 43500, g % 44000, g % 44500, g %
45000,
    g % 45500, g % 46000, g % 46500, g % 47000, g % 47500, g % 48000, g % 48500, g % 49000, g % 49500, g %
50000
FROM generate_series(1,50000000) g;

VACUUM (FREEZE, ANALYZE) perf_row;
VACUUM (FREEZE, ANALYZE) perf_columnar;

```

Para esses dados, você pode ver uma taxa de compactação melhor que 8X na tabela colunar.

```

SELECT pg_total_relation_size('perf_row')::numeric/
    pg_total_relation_size('perf_columnar') AS compression_ratio;
compression_ratio
-----
8.0196135873627944
(1 row)

```

Exemplo

O armazenamento colunar funciona bem com particionamento de tabela. Para ver um exemplo, confira a

documentação da comunidade da Citus Engine, [Como arquivar com armazenamento colunar](#).

Armadilhas

- O armazenamento colunar compacta por faixa. As faixas são criadas por transação, ou seja, a inserção de uma linha por transação colocará linhas individuais em suas próprias faixas. A compactação e o desempenho de faixas de uma única linha serão piores do que uma tabela de linhas. Insira sempre em massa em uma tabela colunar.
- Se cometer um erro e colunarizar várias linhas pequenas, você não poderá desfazê-lo. O único remédio é criar outra tabela colunar e copiar dados da original em uma transação:

```
BEGIN;
CREATE TABLE foo_compacted (LIKE foo) USING columnar;
INSERT INTO foo_compacted SELECT * FROM foo;
DROP TABLE foo;
ALTER TABLE foo_compacted RENAME TO foo;
COMMIT;
```

- Fundamentalmente, dados não compactáveis podem ser um problema, embora o armazenamento colunar ainda possa ser útil na escolha de colunas específicas. Não é necessário carregar as outras colunas na memória.
- Em uma tabela particionada com uma mistura de partições de coluna e linha, as atualizações precisam ser direcionadas com cuidado. Filtre-as para alcançar apenas as partições de linha.
 - Se a operação for direcionada para uma partição de linha específica (por exemplo, `UPDATE p2 SET i = i + 1`), ela será bem-sucedida; se for direcionada a uma partição colunar específica (por exemplo, `UPDATE p1 SET i = i + 1`), ela falhará.
 - Se a operação for direcionada à tabela particionada e houver uma cláusula WHERE que exclui todas as partições colunares (por exemplo, `UPDATE parent SET i = i + 1 WHERE timestamp = '2020-03-15'`), ela será bem-sucedida.
 - Se a operação for direcionada à tabela particionada, mas não filtrar as colunas de chave da partição, ela falhará. Mesmo que haja cláusulas WHERE que correspondam a colunas apenas em partições colunares, não é suficiente; a chave de partição também precisa ser filtrada.

Limitações

Esse recurso ainda tem limitações significativas. Confira [Limites e limitações da Hiperescala \(Citus\)](#).

Próximas etapas

- Confira um exemplo de armazenamento colunar em um [tutorial de série temporal](#) da Citus (link externo).

Banco de dados do Azure para PostgreSQL – opções de configuração do Citus (hiperescala)

09/08/2021 • 3 minutes to read

Computação e armazenamento

Você pode selecionar as configurações de computação e armazenamento independentemente para nós de trabalho e o nó coordenador em um grupo de servidores de hiperescala (Citus). Os recursos de computação são fornecidos como vCores, que representam a CPU lógica do hardware subjacente. O tamanho do armazenamento para provisionamento refere-se à capacidade disponível para os nós coordenadores e de trabalho no seu grupo de servidores de hiperescala (Citus). O armazenamento é usado para arquivos de banco de dados, arquivos temporários, logs de transações e logs do servidor PostgreSQL.

Camada padrão

RECURSO	NÓ DE TRABALHO	NÓ COORDENADOR
Computação, vCores	4, 8, 16, 32, 64	4, 8, 16, 32, 64
Memória por vCore, GiB	8	4
Tamanho do armazenamento, TiB	0,5, 1, 2	0,5, 1, 2
Tipo de armazenamento	Uso Geral (SSD)	Uso Geral (SSD)
IOPS	Até 3 IOPS/GiB	Até 3 IOPS/GiB

A quantidade total de RAM em um único nó de hiperescala (Citus) baseia-se no número selecionado de vCores.

VCORES	UM NÓ DE TRABALHO, GIB RAM	NÓ COORDENADOR, GIB DE RAM
4	32	16
8	64	32
16	128	64
32	256	128
64	432	256

A quantidade total de armazenamento que você provisiona também define a capacidade de entrada/saída disponível para cada nó coordenador e de trabalho.

TAMANHO DO ARMAZENAMENTO, TIB	IOPS MÁXIMO
0,5	1.536
1	3.072

TAMANHO DO ARMAZENAMENTO, TIB	IOPS MÁXIMO
2	6,148

Para todo o cluster de hiperescala (Citus), o IOPS agregado trabalha com os seguintes valores:

NÓS DE TRABALHO	0,5 TIB DE IOPS TOTAL	1 TIB DE IOPS TOTAL	2 TIB DE IOPS TOTAL
2	3.072	6.144	12.296
3	4.608	9.216	18.444
4	6.144	12.288	24.592
5	7.680	15.360	30.740
6	9.216	18.432	36.888
7	10.752	21.504	43.036
8	12.288	24.576	49.184
9	13.824	27.648	55.332
10	15.360	30.720	61.480
11	16.896	33.792	67.628
12	18.432	36.864	73.776
13	19.968	39.936	79.924
14	21.504	43.008	86.072
15	23.040	46.080	92.220
16	24.576	49.152	98.368
17	26.112	52.224	104.516
18	27.648	55.296	110.664
19	29.184	58.368	116.812
20	30.720	61.440	122.960

Camada básica (versão prévia)

IMPORTANT

A camada básica de Hiperescala (Citus) está atualmente em versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

Você pode ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

A [camada básica](#) da Hiperescala (Citus) é um grupo de servidores com apenas um nó. Como não há uma distinção entre os nós de coordenador e de trabalho, não é muito complicado escolher os recursos de computação e armazenamento.

RECURSO	OPÇÕES DISPONÍVEIS
Computação, vCores	2, 4, 8
Memória por vCore, GiB	4
Tamanho de armazenamento, GiB	128, 256, 512
Tipo de armazenamento	Uso Geral (SSD)
IOPS	Até 3 IOPS/GiB

A quantidade total de RAM em um único nó de hiperescala (Citus) baseia-se no número selecionado de vCores.

VCORES	GIB RAM
2	8
4	16
8	32

A quantidade total de armazenamento que você provisiona também define a capacidade disponível para o nó de camada básica.

TAMANHO DE ARMAZENAMENTO, GIB	IOPS MÁXIMO
128	384
256	768
512	1.536

Regiões

Os grupos de servidores de hiperescala (Citus) estão disponíveis nas seguintes regiões do Azure:

- Américas:
 - Sul do Brasil
 - Canadá Central
 - Centro dos EUA

- Leste dos EUA *
- Leste dos EUA 2
- Centro-Norte dos EUA
- Oeste dos EUA 2
- Pacífico Asiático:
 - Leste da Austrália
 - Leste do Japão
 - Coreia Central
 - Sudeste Asiático
- Europa:
 - França Central
 - Norte da Europa
 - Norte da Suíça
 - Sul do Reino Unido
 - Europa Ocidental

(* = dá suporte a [versão prévia dos recursos](#))

Algumas dessas regiões podem não estar inicialmente ativadas em todas as assinaturas do Azure. Se você quiser usar uma região da lista acima e não a vir na sua assinatura, ou se quiser usar uma região que não esteja nessa lista, abra uma [solicitação de suporte](#).

Preços

Para as informações mais recentes sobre preços, consulte a [página de preços](#) do serviço. Para ver os custos da configuração desejada, o [Portal do Azure](#) mostra o custo mensal na guia **Configurar** com base nas opções que você seleciona. Se você não tiver uma assinatura do Azure, poderá usar a calculadora de preços do Azure para obter um preço estimado. No site da [Calculadora de preços do Azure](#), selecione **Adicionar itens**, expanda a categoria **Bancos de dados** e escolha **Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)** para personalizar as opções.

Próximas etapas

Saiba como [criar um grupo de servidores de Hiperescala \(Citus\) no portal](#).

Banco de Dados do Azure para PostgreSQL – Limites e limitações de Hiperescala (Citus)

21/05/2021 • 3 minutes to read

A seção a seguir descreve a capacidade e os limites funcionais no serviço de hiperescala (Citus).

Número máximo de conexões

Cada conexão PostgreSQL (até mesmo as ociosas) usa pelo menos 10 MB de memória, portanto, é importante limitar conexões simultâneas. Eis os limites que escolhemos para manter os nós íntegros:

- Nó coordenador
 - Número máximo de conexões: 300
 - Número máximo de conexões de usuário: 297
- Nó de trabalho
 - Número máximo de conexões: 600
 - Número máximo de conexões de usuário: 597

NOTE

Em um grupo de servidores com [recursos de visualização](#) habilitados, os limites de conexão para o coordenador são ligeiramente diferentes:

- Conexões máximas do nó coordenador
 - 300 para 0 a 3 vCores
 - 500 para 4 a 15 vCores
 - 1000 para 16 ou mais vCores

As tentativas de conexão que ultrapassem esses limites falharão com um erro. O sistema reserva três conexões para nós de monitoramento, e é por isso que há três conexões a menos disponíveis para consultas de usuário do que o total de conexões.

Pool de conexões

O estabelecimento de novas conexões leva tempo. Isso funciona com a maioria dos aplicativos, que solicita muitas conexões de curta duração. É recomendável usar um pooler de conexões para reduzir as transações ociosas e reutilizar as conexões existentes. Para saber mais, acesse nossa [postagem no blog](#).

Você pode executar seu próprio pooler de conexão ou usar o PgBouncer gerenciado pelo Azure.

PgBouncer gerenciado (visualização)

IMPORTANT

O pooler de conexão PgBouncer gerenciado no Hiperescala (Citus) está em visualização no momento. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

Você pode ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

Os poolers de conexão, como o PgBouncer, permitem que mais clientes se conectem ao nó coordenador de uma

só vez. Os aplicativos se conectam ao pooler, e o pooler retransmite comandos para o banco de dados de destino.

Quando os clientes se conectam por meio do PgBouncer, o número de conexões que podem ser executadas ativamente no banco de dados não é alterado. Em vez disso, o PgBouncer enfileira as conexões em excesso e as executa quando o banco de dados está pronto.

A Hiperescala (Citus) agora está oferecendo uma instância gerenciada do PgBouncer para grupos de servidores (em visualização). Ele dá suporte a até 2 mil conexões de cliente simultâneas. Para se conectar por meio do PgBouncer, siga estas etapas:

1. Acesse a página **Cadeias de conexão** do seu grupo de servidores no portal do Azure.
2. Habilite a caixa de seleção **C** do PgBouncer. (as cadeias de conexão listadas serão alteradas.)
3. Atualize os aplicativos cliente para se conectar com a nova cadeia de caracteres.

Dimensionamento de armazenamento

O armazenamento em nós de coordenador e de trabalho pode ser escalado verticalmente (aumentado), mas não pode ser reduzido verticalmente (diminuído).

Tamanho de armazenamento

Há suporte para até 2 TiB de armazenamento em nós de coordenador e de trabalho. Confira as opções de armazenamento disponíveis e o cálculo de IOPS [acima](#) para ver os tamanhos de nó e de cluster.

Criação do banco de dados

O portal do Azure fornece credenciais para se conectar a exatamente um banco de dados por grupo de servidores de Hiperescala (Citus), o banco de dados `citus`. A criação de outro banco de dados não é permitida no momento, e o comando CREATE DATABASE falhará com um erro.

Armazenamento em coluna

Atualmente, a Hiperescala (Citus) tem as seguintes limitações com [tabelas de coluna](#):

- A compactação está no disco, não na memória
- Somente acréscimo (sem suporte a UPDATE/DELETE)
- Sem recuperação de espaço (por exemplo, transações revertidas ainda podem consumir espaço em disco)
- Sem suporte a índice, exames de índice ou exames de índice de bitmap
- Sem tidscans
- Nenhum exemplo de exame
- Não há suporte ao sistema (valores grandes com suporte embutido)
- Não há suporte para instruções ON CONFLICT (exceto ações DO NOTHING sem nenhum destino especificado).
- Não há suporte para bloqueios de tupla (SELECT ... FOR SHARE, SELECT ... FOR UPDATE)
- Não há suporte para o nível de isolamento serializável
- Suporte para o servidor PostgreSQL, somente versões 12+
- Não há suporte para chaves estrangeiras, restrições exclusivas ou restrições de exclusão
- Não há suporte para decodificação lógica
- Não há suporte para exames paralelos dentro do nó
- Não há suporte para gatilhos AFTER ... FOR EACH ROW
- Nenhuma tabela de coluna UNLOGGED

- Nenhuma tabela de coluna TEMPORARY

Próximas etapas

Saiba como [criar um grupo de servidores de Hiperescala \(Citus\) no portal](#).

Pagar antecipadamente pelos recursos de computação de Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus) com capacidade reservada

09/08/2021 • 6 minutes to read

O Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus) agora ajuda você a economizar com o pagamento antecipado de recursos de computação em comparação com o preço pago conforme o uso. Com a capacidade reservada de Hiperescala (Citus), você assume um compromisso inicial com um grupo de servidores Hiperescala (Citus) por um período de um ou três anos para obter um desconto significativo nos custos de computação. Para adquirir a capacidade reservada do Hiperescala (Citus), você precisa especificar a região do Azure, o termo de reserva e a frequência de cobrança.

IMPORTANT

Este artigo é sobre a capacidade reservada para o Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus). Para obter informações sobre a capacidade reservada para o Banco de Dados do Azure para PostgreSQL – Servidor único, consulte [Pagar antecipadamente pelos recursos de computação do Banco de Dados do Azure para PostgreSQL - Servidor único com capacidade reservada](#).

Você não precisa atribuir a reserva a grupos de servidores Hiperescala (Citus) específicos. Um grupo de servidores Hiperescala (Citus) que já está em execução ou aqueles que foram recém-implantados automaticamente têm o benefício de preços reservados. Ao comprar uma reserva, você está pagando antecipadamente os custos de computação por um período de um ou três anos. Assim que você compra uma reserva, os preços de computação de Hiperescala (Citus) que correspondem aos atributos de reserva não são mais cobrados como pagamento conforme o uso.

Uma reserva não cobre custos de software, rede ou armazenamento associados aos grupos de servidores Hiperescala (Citus). No final do período de reserva, o benefício de cobrança expira e os grupos de servidores Hiperescala (Citus) são cobrados pelo preço de pré-pagamento. As reservas não são renovadas automaticamente. Para obter informações sobre preços, consulte a [oferta de capacidade reservada do Banco de Dados do Azure para PostgreSQL – Hiperescala \(Citus\)](#).

Você pode comprar capacidade reservada do Hiperescala (Citus) no [Portal do Azure](#). Pague pela reserva **antecipadamente ou com pagamentos mensais**. Para comprar a capacidade reservada:

- Você deve ter a função de Proprietário em pelo menos uma assinatura de EA (Contrato Enterprise) ou individual com tarifas de pagamento conforme o uso.
- Para as assinaturas de Contrato Enterprise, a opção **Adicionar instâncias reservadas** deve estar habilitada no [Portal do EA](#). Ou, se essa configuração estiver desabilitada, você deve ser um administrador de Contrato Enterprise na assinatura.
- Para o programa Provedor de Solução de Nuvem (CSP), apenas os agentes admin ou agentes de vendas podem adquirir a capacidade reservada de Hiperescala (Citus).

Para obter informações sobre como os clientes de Contrato Enterprise e clientes pagos conforme o uso são cobrados por compras de reserva, consulte:

- [Entender o uso de reserva do Azure para sua inscrição no Contrato Enterprise](#)

- Entender o uso de reserva do Azure para sua assinatura de pague conforme o uso

Determinar o tamanho correto do grupo de servidores antes da compra

O tamanho da reserva é baseado na quantidade total de computação usada pelo nó coordenador e nós de trabalho existentes ou em breve para serem implantados em grupos de servidores Citus (Hiperescala) em uma região específica.

Por exemplo, vamos supor que você esteja executando um grupo de servidores de Hiperescala (Citus) com um nó coordenador 16 vCore e três nós de trabalho 8 vCore. Além disso, vamos supor que você planeja implantar no mês seguinte um grupo de servidores adicional de Hiperescala (Citus) com um nó coordenador 32 vCore e dois nós de trabalho 4 vCore. Vamos supor também que você precise desses recursos por pelo menos um ano.

Nesse caso, compre uma reserva de um ano para:

- Total de 16 vCores + 32 vCores = 48 vCores para nós coordenadores
- Total de 3 nós x 8 vCores + 2 nós x 4 vCores = 24 + 8 = 32 vCores para nós de trabalho

Comprar capacidade reservada do Banco de Dados do Azure para PostgreSQL

1. Entre no [portal do Azure](#).
2. Selecione **Todos os serviços > Reservas**.
3. Selecione **Adicionar**. No painel **Reservas de compra**, selecione **Banco de Dados do Azure para PostgreSQL** para comprar uma nova reserva para seus bancos de dados PostgreSQL.
4. Selecione o tipo **Computação de Hiperescala (Citus)** a ser comprado e clique em **Selecionar**.
5. Revise a quantidade do tipo de computação selecionado na guia **Produtos**.
6. Continue na guia **Comprar + Revisar** para concluir sua compra.

A tabela a seguir descreve os campos obrigatórios.

CAMPO	DESCRIÇÃO
Subscription	A assinatura usada para a reserva de capacidade reservada do Banco de Dados do Azure para PostgreSQL. O método de pagamento na assinatura é cobrado pelos custos iniciais da reserva de capacidade reservada do Banco de Dados do Azure para PostgreSQL. O tipo de assinatura deve ser um contrato corporativo (números de oferta MS-AZR-0017P ou MS-AZR-0148P) ou um contrato individual de pagamento conforme o uso (números de oferta MS-AZR-0003P ou MS-AZR-0023P). Para uma assinatura do Contrato Enterprise, os encargos são deduzidos do saldo do Pagamento antecipado do Azure da inscrição (chamado anteriormente de compromisso monetário) ou cobrados como excedente. Para uma assinatura individual com taxas pagas conforme o uso, as cobranças são feitas na forma de pagamento de cartão de crédito ou de fatura na assinatura.

CAMPO	DESCRIÇÃO
Escopo	O escopo da reserva vCore pode cobrir uma assinatura ou várias assinaturas (escopo compartilhado). Se você selecionar Compartilhado , o desconto de reserva de vCore será aplicado aos grupos de servidores de Hiperescala (Citus) em execução em qualquer assinatura no contexto de cobrança. Para clientes do Contrato Enterprise, o escopo compartilhado é o registro e inclui todas as assinaturas no registro. Para clientes de pagamento conforme o uso, o escopo compartilhado consiste em todas as assinaturas de pagamento conforme o uso criadas pelo administrador da conta. Se você selecionar Assinatura única , o desconto de reserva de vCore será aplicado aos grupos de servidores de Hiperescala (Citus) nessa assinatura. Se você selecionar Um único grupo de recursos , o desconto de reserva será aplicado aos grupos de servidores de Hiperescala (Citus) na assinatura selecionada e ao grupo de recursos selecionado nessa assinatura.
Região	A região do Azure abrangida pela reserva de capacidade reservada do Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus).
Termo	Um ano ou três anos.
Quantidade	A quantidade de recursos de computação que estão sendo comprados na reserva de capacidade reservada de Hiperescala (Citus). Em particular, o número de vCores do nó coordenador ou de trabalho na região do Azure selecionada que estão sendo reservados e que obterão o desconto de cobrança. Por exemplo, se você estiver executando (ou planeja executar) grupos de servidores de Hiperescala (Citus) com a capacidade de computação total de 64 vCores de nó coordenador e 32 vCores de nó de trabalho na região Leste dos EUA, especifique a quantidade como 64 e 32 para nós coordenador e de trabalho, respectivamente, para maximizar o benefício para todos os servidores.

Cancelar, trocar ou reembolsar reservas

É possível cancelar, trocar ou reembolsar reservas com determinadas limitações. Para obter mais informações, confira [Trocas e reembolsos via autoatendimento para reservas do Azure](#).

Flexibilidade de tamanho do vCore

A flexibilidade de tamanho do vCore ajuda você a escalar verticalmente ou horizontalmente nós coordenadores e de trabalho dentro de uma região, sem perder o benefício de capacidade reservada.

Precisa de ajuda? Fale conosco

Se você tiver dúvidas ou precisar de ajuda, [crie uma solicitação de suporte](#).

Próximas etapas

O desconto de reserva de vCore é aplicado automaticamente ao número de grupos de servidores de Hiperescala (Citus) que correspondem aos atributos e ao escopo de reserva de capacidade reservada do Banco de Dados PostgreSQL do Azure. Você pode atualizar o escopo da reserva de capacidade reservada do Banco de

Dados PostgreSQL do Azure – Hiperescala (Citus) por meio do portal do Azure, PowerShell, CLI do Azure ou da API.

Para saber mais sobre reservas do Azure, confira os seguintes artigos:

- [O que são reservas do Azure?](#)
- [Gerenciar reservas do Azure](#)
- [Entender o desconto de reserva do Azure](#)
- [Entender o uso de reserva para a sua assinatura paga conforme o uso](#)
- [Entender o uso de reserva para sua inscrição no Contrato Enterprise](#)
- [Reservas do Azure no programa de Provedor de Soluções na Nuvem da Central de Parceiros](#)

Versões de banco de dados com suporte no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 2 minutes to read

Versões do PostgreSQL

IMPORTANT

Versões personalizáveis do PostgreSQL no Hiperescala (Citus) estão atualmente em versão prévia. Essa versão prévia é fornecida sem um Contrato de Nível de Serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

É possível ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

A versão do PostgreSQL em execução em um grupo de servidores de Hiperescala (Citus) é personalizável durante a criação. Escolher algo diferente da versão 11 é atualmente uma versão prévia do recurso.

O Hiperescala (Citus) atualmente dá suporte às seguintes versões principais:

PostgreSQL versão 13 (versão prévia)

A versão secundária atual é 13.2. Consulte a [documentação do PostgreSQL](#) para saber mais sobre aperfeiçoamentos e correções nesta versão secundária.

PostgreSQL versão 12 (versão prévia)

A versão secundária atual é 12.6. Consulte a [documentação do PostgreSQL](#) para saber mais sobre aperfeiçoamentos e correções nesta versão secundária.

PostgreSQL versão 11

A versão secundária atual é 11.11. Consulte a [documentação do PostgreSQL](#) para saber mais sobre aperfeiçoamentos e correções nesta versão secundária.

PostgreSQL versão 10 e anteriores

Não há suporte para PostgreSQL versão 10 e anteriores para o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus).

Citus e outras versões de extensão

Dependendo da versão do PostgreSQL em execução em um grupo de servidores, diferentes [versões das extensões do Postgres](#) também serão instaladas. Em particular, Postgres 13 vem com o Citus 10, e versões anteriores do Postgres vêm com Citus 9.5.

Próximas etapas

- Veja quais [extensões](#) estão instaladas em quais versões.
- Saiba como [criar um grupo de servidores de Hiperescala \(Citus\)](#).

Extensões no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 6 minutes to read

O PostgreSQL fornece a capacidade de estender a funcionalidade do banco de dados usando extensões. As extensões permitem o agrupamento de vários objetos SQL relacionados em um único pacote que pode ser carregado ou removido de seu banco de dados com um único comando. Após serem carregadas no banco de dados, as extensões funcionam como recursos internos. Para saber mais sobre extensões PostgreSQL, consulte [Empacotar objetos relacionados em uma extensão](#).

Usar extensões do PostgreSQL

As extensões PostgreSQL devem ser instaladas no banco de dados para que você possa usá-las. Para instalar uma extensão específica, execute o comando [CREATE EXTENSION](#) na ferramenta psql para carregar os objetos empacotados em seu banco de dados.

No momento, o Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus) dá suporte a um subconjunto de extensões de chave, conforme listado abaixo. Não há suporte para extensões que não aquelas listadas. Não é possível criar sua própria extensão com o Banco de Dados do Azure para o serviço PostgreSQL.

Extensões com suporte do Banco de Dados do Azure para PostgreSQL

As tabelas a seguir listam as extensões PostgreSQL padrão que têm suporte atualmente do Banco de Dados do Azure para PostgreSQL. Essas informações também estão disponíveis por meio da execução de

```
SELECT * FROM pg_available_extensions;
```

Extensões de tipos de dados

EXTENSÃO	DESCRIÇÃO
citext	Fornece um tipo de cadeia de caracteres que não diferencia maiúsculas de minúsculas.
cube	Fornece um tipo de dados para cubos multidimensionais.
hll	Fornece uma estrutura de dados HyperLogLog.
hstore	Fornece um tipo de dados para armazenar conjuntos de pares chave-valor.
isn	Fornece tipos de dados para padrões de numeração de produto internacionais.
lo	Manutenção de Objeto Grande.
ltree	Fornece um tipo de dados para estruturas semelhantes a árvores hierárquicas.

EXTENSÃO	DESCRIÇÃO
seg	Tipo de dados para representar segmentos de linha ou intervalos de ponto flutuante.
tdigest	Tipo de dados para acumulação online de estatísticas baseadas em classificação, como médias quantil e truncada.
topn	Tipo para top-n JSONB.

Extensões de pesquisa de texto completo

EXTENSÃO	DESCRIÇÃO
dict_int	Fornece um modelo de dicionário de pesquisa de texto para números inteiros.
dict_xsyn	Modelo de dicionário de pesquisa de texto para processamento estendido de sinônimo.
unaccent	Um dicionário de pesquisa de texto que remove acentos (sinais diacríticos) dos lexemas.

Extensões de funções

EXTENSÃO	DESCRIÇÃO
autoinc	Funções para incrementar campos automaticamente.
earthdistance	Fornece um meio para calcular as distâncias ortodrônicas na superfície da Terra.
fuzzystrmatch	Fornece várias funções para determinar semelhanças e a distância entre cadeias de caracteres.
insert_username	Funções para controlar quem alterou uma tabela.
intagg	Agregador inteiro e enumerador (obsoleto).
intarray	Fornece funções e operadores para manipular matrizes de inteiros sem nulos.
moddatetime	Funções para controlar a hora da última modificação.
pg_partman	Gerencia as tabelas de partição por hora ou ID.
pg_trgm	Fornece funções e operadores para determinar a semelhança de texto alfanumérico, com base na correspondência de trigram.
pgcrypto	Fornece funções de criptografia.
refint	Funções para implementar a integridade referencial (obsoleto).

EXTENSÃO	DESCRIÇÃO
análise de sessão _	Funções para consultar matrizes hstore.
tablefunc	Fornece funções que manipulam tabelas inteiras, incluindo a tabela de referência cruzada.
tcn	Notificações de alteração disparadas.
timetravel	Funções para implementação de viagem de tempo.
uuid-ossp	Gera UUIDs (identificadores exclusivos universais).

Extensões de Hiperescala (Citus)

EXTENSÃO	DESCRIÇÃO
citus	Banco de dados distribuído Citus.

Extensões de tipos de índice

EXTENSÃO	DESCRIÇÃO
bloom	Método de acesso bloom – índice baseado em arquivo de assinatura.
btree_gin	Fornece classes de operador GIN de exemplo que implementam um comportamento como da árvore B para certos tipos de dados.
btree_gist	Fornece classes de operador de índice GiST que implementam a árvore B.

Extensões de linguagem

EXTENSÃO	DESCRIÇÃO
plpgsql	Linguagem de procedimento carregável PL/pgSQL.

Extensões diversas

EXTENSÃO	DESCRIÇÃO
adminpack	Funções administrativas para PostgreSQL.
amcheck	Funções para verificar a integridade da relação.
dblink	Um módulo que suporta conexões com outros bancos de dados do PostgreSQL de dentro de uma sessão de banco de dados. Consulte a seção "dblink e postgres_fdw" para obter informações sobre essa extensão.
file_fdw	Wrapper de dados estrangeiros para acesso de arquivo simples.

EXTENSÃO	DESCRIÇÃO
pageinspect	Inspecione o conteúdo das páginas do banco de dados em um nível baixo.
pg_buffercache	Fornece um meio para examinar o que está acontecendo no cache do buffer compartilhado em tempo real.
pg_cron	Agendador de trabalhos para PostgreSQL.
pg_freespacemap	Examina o mapa de espaço livre (FSM).
pg_prewarm	Fornece uma maneira para carregar dados de relação no cache de buffer.
pg_stat_statements	Fornece um meio para rastrear as estatísticas de execução de todas as instruções SQL executadas por um servidor. Consulte a seção "pg_stat_statements" para obter informações sobre essa extensão.
pg_visibility	Examine a VM (mapa de visibilidade) e as informações de visibilidade no nível da página.
pgrowlocks	Fornece um meio para mostrar informações de bloqueio de nível de linha.
pgstattuple	Fornece um meio para mostrar estatísticas em nível de tupla.
postgres_fdw	Wrapper de dados externos usado para acessar dados armazenados em servidores PostgreSQL externos. Consulte a seção "dblink e postgres_fdw" para obter informações sobre essa extensão.
sslinfo	Informações sobre certificados TLS/SSL.
tsm_system_rows	Método TABLESAMPLE, que aceita o número de linhas como um limite.
tsm_system_time	Método TABLESAMPLE, que aceita o tempo em milissegundos como um limite.
xml2	Consulta XPath e XSLT.

Extensões PostGIS

EXTENSÃO	DESCRIÇÃO
PostGIS , postgis_topology, postgis_tiger_geocoder, postgis_sfsgal	Objetos espaciais e geográficos para PostgreSQL.
address_standardizer, address_standardizer_data_us	Usado para analisar um endereço em elementos constituintes. Usado para oferecer suporte à etapa de normalização de endereços de geocodificação.
postgis_sfsgal	Funções SFCGAL do PostGIS.

EXTENSÃO	DESCRIÇÃO
postgis_tiger_geocoder	Geocodificador PostGIS Tiger e geocodificador reverso.
postgis_topology	Tipos e funções espaciais da topologia do PostGIS.

pg_stat_statements

A extensão [pg_stat_statements](#) é pré-carregada em cada servidor do Banco de Dados do Azure para PostgreSQL para fornecer a você uma maneira de acompanhar estatísticas de execução de instruções SQL.

A configuração `pg_stat_statements.track` controla quais instruções são contadas pela extensão. Ele usa como padrão `top`, o que significa que todas as instruções emitidas diretamente pelos clientes são controladas. Dois outros níveis de rastreamento são `none` e `all`. Essa definição é configurável como um parâmetro de servidor por meio de [portal do Azure](#) ou da [CLI do Azure](#).

Há um equilíbrio entre as informações de execução de consulta fornecida por `pg_stat_statements` e o impacto no desempenho do servidor, que regista cada instrução SQL. Se você não está usando ativamente a extensão `pg_stat_statements`, recomendamos que você defina `pg_stat_statements.track` como `none`. Alguns serviços de monitoramento de terceiros podem depender de `pg_stat_statements` para fornecer informações de desempenho de consulta; portanto, confirme se este é o caso para você ou não.

dblink e postgres_fdw

Com `postgres_fdw`, você pode se conectar de um servidor PostgreSQL a outro ou a outro banco de dados no mesmo servidor. O servidor de recebimento precisa permitir conexões do servidor de envio por meio de seu firewall. Para usar essas extensões para se conectar entre os servidores do banco de dados do Azure para servidor PostgreSQL ou grupos de servidores Citus (hiperescala), defina **Permitir que os serviços e recursos do Azure accessem este grupo de servidores (ou servidor)** como ativado. Você também precisará ativar essa configuração se quiser usar as extensões para fazer um loop de volta para o mesmo servidor. A configuração **Permitir que os serviços e recursos do Azure accessem este grupo de servidores** pode ser encontrada na página portal do Azure para o grupo de servidores de Hiperescala (Citus) em **Rede**. Atualmente, não há suporte para conexões de saída do banco de dados do Azure para PostgreSQL de servidor único e Hiperescala (Citus), exceto para conexões com outros grupos de servidores do banco de dados do Azure para servidor PostgreSQL e de hiperescala (Citus).

Escolher o tamanho inicial para o grupo de servidores Hiperescala (Citus)

09/08/2021 • 2 minutes to read

O tamanho de um grupo de servidores, seu número de nós e sua capacidade de hardware, é [fácil de alterar](#)). No entanto, você ainda precisa escolher um tamanho inicial para um novo grupo de servidores. Aqui estão algumas dicas para uma boa escolha.

Casos de uso

A Hiperescala (Citus) é usada frequentemente das maneiras a seguir.

SaaS multilocatário

Quando você estiver migrando para o Hiperescala (Citus) de uma instância de banco de dados PostgreSQL de nó único existente, escolha um cluster em que o número de vCores de trabalho e a RAM, no total, seja igual ao da instância original. Nesses cenários, vimos melhorias de desempenho 2-3x porque a fragmentação melhora a utilização de recursos, permitindo índices menores etc.

A contagem vCores é, na verdade, a única decisão. A alocação de RAM é determinada no momento com base na contagem de vCores, conforme descrito na página [Opções de configuração do Hiperescala \(Citus\)](#). O nó coordenador não exige tanta RAM quanto os trabalhadores, mas não há como escolher RAM e vCores de forma independente.

Análise em tempo real

Total de vCores: quando os dados de trabalho se ajustam à RAM, você pode esperar uma melhoria de desempenho linear no Hiperescala (Citus) proporcional ao número de núcleos de trabalho. Para determinar o número certo de vCores para suas necessidades, considere a latência atual para consultas no banco de dados de nó único e a latência necessária no Hiperescala (Citus). Divida a latência atual pela latência desejada e arredonde o resultado.

RAM de Trabalho: o melhor caso seria fornecer memória suficiente para que a maior parte do conjunto de trabalho caiba na memória. O tipo de consultas que seu aplicativo usa afetará os requisitos de memória. Você pode executar EXPLAIN ANALYZE em uma consulta para determinar a quantidade de memória necessária.

Lembre-se de que vCores e RAM são dimensionados juntos, conforme descrito no artigo [Opções de configuração do Hiperescala \(Citus\)](#).

Como escolher uma camada da Hiperescala (Citus)

IMPORTANT

A camada básica de Hiperescala (Citus) está atualmente em versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

Você pode ver a lista completa dos novos recursos em [versão prévia dos recursos para Hiperescala \(Citus\)](#).

As seções acima dão uma ideia do número de vCores e da quantidade de RAM necessária para cada caso de uso. Você pode atender a essas demandas escolhendo entre duas camadas da Hiperescala (Citus): a camada básica e a camada standard.

A camada básica usa um único nó de banco de dados para executar o processamento, enquanto a camada standard permite mais nós. Fora isso, as camadas são idênticas, oferecendo os mesmos recursos. Em alguns casos, os vCores e o espaço em disco de um único nó podem ser dimensionados para o tamanho necessário e, em outros casos, eles requerem a cooperação de vários nós.

Para obter uma comparação das camadas, confira a página de conceitos da [camada básica](#).

Próximas etapas

- [Escalar um grupo de servidores](#)
- Saiba mais sobre as [opções de desempenho](#) do grupo de servidores.

Criar um grupo de servidores Hiperescala (Citus)

09/08/2021 • 2 minutes to read

Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus) fornece dimensionamento de autoatendimento para lidar com o aumento da carga. O portal do Azure facilita a adição de nós de trabalho, o aumento de vCores e o armazenamento de nós existentes. A adição de nós não causa nenhum tempo de inatividade e até mesmo mover fragmentos para os novos nós (chamado de [rebalanceamento de fragmentos](#)) ocorre sem interromper consultas.

Adicionar nós de trabalho

Para adicionar nós, acesse a guia **Computação + armazenamento** em seu grupo de servidores de Hiperescala (Citus). Arrastar o controle deslizante para a contagem de **Nós de trabalho** vai alterar o valor.

NOTE

Um grupo de servidores de Hiperescala (Citus) criado com a [camada básica \(versão prévia\)](#) não possui trabalhadores. Aumentar a contagem de trabalhadores automaticamente gradua o grupo de servidores para a camada Standard. Depois de graduar um grupo de servidores para a camada Standard, não é possível fazer downgrade de volta para a camada básica.

The screenshot shows the Azure portal interface for managing a Citus cluster. On the left, there's a sidebar with navigation links: Search (Cmd+/, Overview, Activity log, Tags, Settings, Compute + storage, Networking, Connection strings, Roles, Coordinator node parameters, Worker node parameters, Locks, Support + troubleshooting, and New support request). The 'Compute + storage' link is currently selected. The main content area has a heading 'You can scale your Hyperscale (Citus) cluster by adding nodes to it without cluster downtime and by scaling compute cores on coordinator and worker nodes.' Below this, under 'Worker nodes', there's a slider for 'Worker node count' set to 2 nodes. A note says 'If you need more than 20 nodes, contact us'. Under 'Configuration (per worker node)', there are sliders for 'vCores' (set to 4) and 'Storage' (set to 0.5 TiB). A warning message 'Storage cannot be scaled down' is shown. To the right, there's a note: 'Your configuration can use up to 3 IOPS / GiB.' Below this, under 'Coordinator node', there's a section for 'Configuration (coordinator node)' with sliders for 'vCores' (set to 4) and 'Storage' (set to 0.5 TiB). A warning message 'Storage cannot be scaled down' is shown. To the right, there's a note: 'Your configuration can use up to 3 IOPS / GiB.'

Clique no botão **Salvar** para fazer com que o valor alterado entre em vigor.

NOTE

Após aumentar e salvar os nós de trabalho, a quantidade deles não poderá ser reduzida com o controle deslizante.

NOTE

Para aproveitar os nós recém-adicionados, é necessário [reequilibrar os fragmentos](#) da tabela distribuída, o que significa mover alguns [fragmentos](#) dos nós existentes para os novos.

Aumentar ou diminuir vCores em nós

Além de adicionar nós, será possível aumentar as funcionalidades dos nós existentes. Ajustar a capacidade de computação para cima e para baixo poderá ser útil para executar testes de desempenho, e alterações de curto ou longo prazo em demandas de tráfego.

Para alterar os vCores de todos os nós de trabalho, ajuste o controle deslizante **vCores** em **Configuração (por nó de trabalho)**. Os vCores do nó coordenador podem ser ajustados independentemente. Ajuste o controle deslizante **vCores** em **Configuração (nó coordenador)**.

Aumentar o armazenamento em nós

Além de adicionar nós, será possível aumentar o espaço em disco dos nós existentes. O aumento do espaço em disco pode permitir que você faça mais com os nós de trabalho existentes antes de precisar adicionar mais nós de trabalho.

Para alterar o armazenamento de todos os nós de trabalho, ajuste o controle deslizante **armazenamento** em **Configuração (por nó de trabalho)**. O armazenamento do nó coordenador pode ser ajustado independentemente. Ajuste o controle deslizante **armazenamento** em **Configuração (nó coordenador)**.

NOTE

Após aumentar e salvar o armazenamento por nó, ele não poderá ser reduzido com o controle deslizante.

Próximas etapas

- Saiba mais sobre as [opções de desempenho](#) do grupo de servidores.
- [Rebalancear fragmentos de tabela distribuídos](#) para que todos os nós de trabalho possam participar de consultas paralelas

Reequilibrar fragmentos no grupo de servidores Hiperescala (Citus)

09/08/2021 • 2 minutes to read

Para aproveitar os nós recém-adicionados, é necessário reequilibrar os [fragmentos](#) da tabela distribuída, o que significa mover alguns fragmentos dos nós existentes para os novos.

Determinar se o grupo de servidores precisa de um reequilíbrio

O portal do Azure pode mostrar se os dados são distribuídos igualmente entre os nós de trabalho em um grupo de servidores. Para vê-lo, vá para a página **Rebalanceador de fragmento** no menu **Gerenciamento de grupo de servidores**. Se os dados forem distorcidos entre os trabalhadores, você verá a mensagem **Recomenda-se o rebalanceamento**, juntamente com uma lista do tamanho de cada nó.

Se os dados já estiverem balanceados, você verá a mensagem **O rebalanceamento não é recomendado no momento**.

Executar o rebalanceador de fragmentos

Para iniciar o rebalanceador de fragmentos, você precisa se conectar ao nó de coordenador do grupo de servidores e executar a função do SQL `rebalance_table_shards` em tabelas distribuídas. A função reequilibra todas as tabelas no grupo de [colocação](#) da tabela nomeada em seu argumento. Portanto, não é necessário chamar a função para cada tabela distribuída, basta chamá-la em uma tabela representativa de cada grupo de colocação.

```
SELECT rebalance_table_shards('distributed_table_name');
```

Monitorar o progresso do rebalanceamento

Para assistir ao rebalanceador depois de iniciá-lo, volte para a portal do Azure. Abra a página **Rebalanceamento de fragmentos** no **Gerenciamento de grupo de servidores**. Ele mostrará a mensagem **O rebalanceamento está em andamento** junto com duas tabelas.

A primeira tabela mostra o número de fragmentos que se entram ou saem de um nó, por exemplo, "6 de 24 que entraram". A segunda tabela mostra o progresso por tabela de banco de dados: nome, contagem de fragmentos afetada, tamanho dos dados afetados e status de rebalanceamento.

Selecione o botão **Atualizar** para atualizar a página. Quando o rebalanceamento for concluído, ele informará novamente **O rebalanceamento não é recomendado no momento**.

Próximas etapas

- Saiba mais sobre as [opções de desempenho](#) do grupo de servidores.
- [Escalar um grupo de servidores](#) vertical ou horizontalmente
- Consulte o material de referência [rebalance_table_shards](#)

Gerenciar regras de firewall para o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

09/08/2021 • 3 minutes to read

As regras de firewall no nível do servidor podem ser usadas para gerenciar o acesso a um nó coordenador de Hiperescala (Citus) de um endereço IP especificado ou de um intervalo de endereços IP.

Pré-requisitos

Para seguir este guia de instruções, você precisa:

- Um grupo de servidores [Criar um grupo de servidores do Banco de Dados do Azure para PostgreSQL – Hiperescala \(Citus\)](#).

Criar uma regra de firewall de nível de servidor no portal do Azure

NOTE

Essas configurações também são acessíveis durante a criação de um grupo de servidores do Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus). Na guia **Rede**, clique em **Ponto de extremidade público**.

Basics **Networking** Tags Review + create

Configure networking access and security for your server group. [Learn more](#)

Network connectivity

You can connect to your server group publicly from public IP addresses you specify or by allowing access for all Azure services and resources.

Connectivity method

Public access

Private access (Virtual network)

1. Na página do grupo de servidores PostgreSQL, no título Segurança, clique em **Rede** para abrir as Regras de firewall.

The screenshot shows the Azure portal interface for managing a server group named 'jonels-test'. The left sidebar includes options like Overview, Activity log, Tags, Settings (Compute + storage, Connection strings, Roles, Coordinator node parameters, Worker node parameters, Locks), Security, Networking (selected), Support + troubleshooting, and New support request.

In the main content area, under 'Firewall rules', it states: 'Inbound connections from the IP addresses specified below will be allowed to port 5432 on the coordinator (jonels-test-c) in your server group (jonels-test).'

A note indicates: 'Connections from the IP addresses specified below provide access to the coordinator jonels-test-c in server group jonels-test.'

An option 'Allow Azure services and resources to access this server group' has 'No' selected. Below this are buttons for '+ Add current client IP address (207.153.12.62)' and '+ Add 0.0.0.0 - 255.255.255.255'.

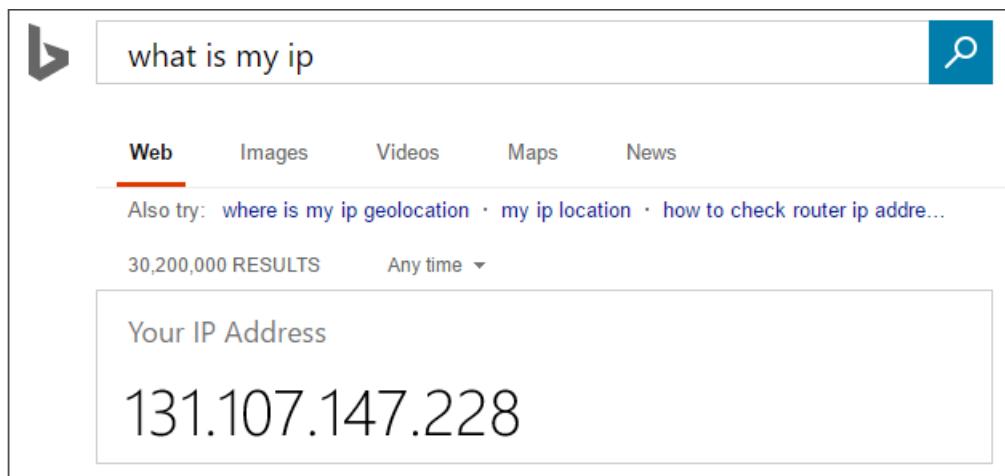
Firewall rule name	Start IP	End IP
Firewall rule name	Start IP	End IP

2. Clique em **Adicionar endereço IP do cliente atual** para criar uma regra de firewall com o endereço IP público do computador, como visto pelo sistema do Azure.

This screenshot is identical to the one above, but the button '+ Add current client IP address (207.153.12.62)' is highlighted with a red box.

Como alternativa, clicar em **+ Adicionar 0.0.0.0 - 255.255.255.255** (à direita da opção B) permite que não apenas o IP, mas a Internet inteira acesse a porta 5432 do nó coordenador. Nessa situação, os clientes ainda precisam fazer logon com o nome de usuário e a senha corretos para usar o cluster. No entanto, recomendamos permitir acesso mundial apenas por períodos curtos e somente para bancos de dados que não são de produção.

3. Verifique seu endereço IP antes de salvar a configuração. Em algumas situações, o endereço IP observado pelo Portal do Azure é diferente do endereço IP usado ao acessar a Internet e os servidores do Azure. Portanto, talvez seja necessário alterar o IP inicial e o IP final para fazer a regra funcionar conforme o esperado. Use um mecanismo de pesquisa ou outra ferramenta online para verificar seu próprio endereço IP. Por exemplo, pesquise "qual é meu IP".



4. Adicionar outros intervalos de endereço. Nas regras de firewall, você pode especificar apenas um endereço IP ou um intervalo de endereços. Se você desejar limitar a regra a um único endereço IP, digite o mesmo endereço no campo IP inicial e IP final. Abrir o firewall permite que administradores, usuários e aplicativos accessem o nó coordenador na porta 5432.
5. Clique em **Salvar** na barra de ferramentas para salvar essa regra de firewall no nível de servidor. Aguarde a confirmação de que a atualização das regras de firewall foi bem-sucedida.

Coneção pelo Azure

Há uma forma fácil de conceder acesso ao banco de dados Hiperescala (Citus) para aplicativos hospedados no Azure (como por exemplo, um aplicativo dos Aplicativos Web do Azure ou aqueles em execução em uma VM do Azure). Basta definir a opção **Permitir que serviços e recursos do Azure accessem este grupo de servidores** como **Sim** no portal, no painel **Rede**, e clicar em **Salvar**.

IMPORTANT

Esta opção configura o firewall para permitir todas as conexões do Azure, incluindo as conexões das assinaturas de outros clientes. Ao selecionar essa opção, verifique se as permissões de logon e de usuário limitam o acesso somente a usuários autorizados.

Gerenciar regras de firewall existentes no nível de servidor pelo Portal do Azure

Repita as etapas para gerenciar as regras de firewall.

- Para adicionar o computador atual, clique em + **Adicionar endereço IP do cliente atual**. Clique em **Salvar** para salvar as alterações.
- Para adicionar mais endereços IP, digite o Nome da Regra, o Endereço IP Inicial e o Endereço IP Final. Clique em **Salvar** para salvar as alterações.
- Para modificar uma regra existente, clique em qualquer um dos campos na regra e modifique. Clique em **Salvar** para salvar as alterações.
- Para excluir uma regra existente, clique nas reticências [...] e clique em **Excluir** para remover a regra. Clique em **Salvar** para salvar as alterações.

Próximas etapas

- Saiba mais sobre o [Conceito de regras de firewall](#), incluindo como solucionar problemas de conexão.

Criar usuários no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

09/08/2021 • 2 minutes to read

A conta do administrador do servidor

O mecanismo PostgreSQL usa [funções](#) para controlar o acesso a objetos de banco de dados, e um grupo de servidores Hiperescala (Citus) recém-criado vem com várias funções predefinidas:

- As [funções padrão do PostgreSQL](#)
- `azure_pg_admin`
- `postgres`
- `citus`

Como Hiperescala (Citus) é um serviço PaaS gerenciado, somente a Microsoft pode entrar com a função `postgres` de superusuário. O Hiperescala (Citus) tem a função `citus` para acesso administrativo limitado.

Permissões para a função `citus`:

- Leia todas as variáveis de configuração, até mesmo as normalmente visíveis somente para superusuários.
- Leia todas as exibições de `__*` e use várias extensões relacionadas a estatísticas – até mesmo exibições ou extensões normalmente visíveis somente para superusuários.
- Execute funções de monitoramento que podem receber bloqueios ACCESS SHARE em tabelas, potencialmente por muito tempo.
- [Crie extensões do PostgreSQL](#) (porque a função é membro de `azure_pg_admin`).

Em especial, a função `citus` tem algumas restrições:

- Não é possível criar funções
- Não é possível criar bancos de dados

Como criar outras funções de usuário

Conforme mencionado, a conta de administrador `citus` não tem permissão para criar usuários adicionais. Para adicionar um usuário, use a interface do portal do Azure.

1. Acesse a página [Funções](#) do grupo de servidores Hiperescala (Citus) e clique em + Adicionar:

Search (Cmd+/) < + Add

Server group roles

New user roles will be created on all nodes in your server group (jonels-test) in addition to the built-in 'citus' role. [Learn more](#)

Name: citus

Settings

- Overview
- Activity log
- Tags
- Compute + storage
- Connection strings
- Roles**
- Coordinator node parameters
- Worker node parameters
- Locks

Security

- Networking

Support + troubleshooting

- New support request

2. Digite o nome e a senha da função. Clique em **Save** (Salvar).

Add role

Save Discard

Role name * ⓘ

Password *

Confirm password *

O usuário será criado no nó coordenador do grupo de servidores e propagado para todos os nós de trabalho. As funções criadas no portal do Azure têm o atributo `LOGIN`, o que significa que elas são usuários verdadeiros que podem entrar no banco de dados.

Como modificar privilégios para a função de usuário

Novas funções de usuário normalmente são usadas para dar acesso ao banco de dados com privilégios restritos. Para modificar privilégios de usuário, use comandos PostgreSQL padrão, usando uma ferramenta como PgAdmin ou psql. (Confira [Conectar-se com psql](#) no início rápido do Hiperescala [Citus].)

Por exemplo, para permitir que `db_user` leia `mytable`, conceda a permissão:

```
GRANT SELECT ON mytable TO db_user;
```

O Hiperescala (Citus) propaga instruções GRANT de tabela única em todo o cluster e as aplica em todos os nós de trabalho, além de propagar GRANTS que são de todo o sistema (por exemplo, para todas as tabelas de um esquema):

```
-- applies to the coordinator node and propagates to workers
GRANT SELECT ON ALL TABLES IN SCHEMA public TO db_user;
```

Como excluir uma função de usuário ou alterar sua senha

Para atualizar um usuário, visite a página **Funções** do grupo de servidores Hiperescala (Citus) e clique nas reticências ... ao lado do usuário. As reticências abrirão um menu para excluir o usuário ou redefinir sua senha.

The screenshot shows the Azure portal interface for managing a Citus server group. The left sidebar includes sections for Overview, Activity log, Tags, Settings (with Compute + storage, Connection strings, Roles selected, Coordinator node parameters, Worker node parameters, and Locks), Security (Networking), Support + troubleshooting (New support request), and a New support request button. The main content area is titled 'Server group roles' and displays a table with one row for the 'citus' role. A context menu is open over this row, showing options: Delete (highlighted with a dashed blue border), Reset the password, and three vertical dots (...).

A função `citus` é privilegiada e não pode ser excluída.

Próximas etapas

Abra o firewall dos endereços IP dos computadores dos novos usuários para habilitar a conexão: [Criar e gerenciar regras de firewall do Hiperescala \(Citus\) no portal do Azure](#).

Para saber mais sobre o gerenciamento da conta de usuário do banco de dados, confira a documentação do produto PostgreSQL:

- [Funções e privilégios de banco de dados](#)
- [Sintaxe GRANT](#)
- [Privilégios](#)

Determinar o tamanho da tabela e da relação

21/05/2021 • 2 minutes to read

A maneira usual de localizar tamanhos de tabelas no PostgreSQL, `pg_total_relation_size`, drasticamente informa de forma errônea o tamanho das tabelas distribuídas em Hiperescala (Citus). Tudo o que essa função faz em um grupo de servidores Hiperescala (Citus) é revelar o tamanho das tabelas no nó coordenador. Na realidade, os dados das tabelas distribuídas residem nos nós de trabalho (em fragmentos) e não no coordenador. Uma medida verdadeira de tamanho de tabelas distribuídas é obtida como uma soma de tamanhos de fragmentos. O Hiperescala (Citus) fornece funções auxiliares para consultar essas informações.

FUNÇÃO	RETORNOS
<code>citus_relation_size(relation_name)</code>	<ul style="list-style-type: none">Tamanho dos dados reais da tabela ("bifurcação principal").Uma relação pode ser o nome de uma tabela ou um índice.
<code>citus_table_size(relation_name)</code>	<ul style="list-style-type: none"><code>citus_relation_size</code> plus:<ul style="list-style-type: none">tamanho do mapa de espaço livretamanho do mapa de visibilidade
<code>citus_total_relation_size(relation_name)</code>	<ul style="list-style-type: none"><code>citus_table_size</code> plus:<ul style="list-style-type: none">tamanho dos índices

Essas funções são análogas a três das [funções de tamanho de objeto](#) PostgreSQL padrão, exceto se não conseguirem se conectar a um nó, quando retornarão erros.

Exemplo

Veja como listar os tamanhos de todas as tabelas distribuídas:

```
SELECT logicalrelid AS name,
       pg_size.pretty(citus_table_size(logicalrelid)) AS size
  FROM pg_dist_partition;
```

Saída:

name	size
github_users	39 MB
github_events	37 MB

Próximas etapas

- Saiba como [dimensionar um grupo de servidores](#) para conter mais dados.
- Distinguir [tipos de tabela](#) em um grupo de servidores Hiperescala (Citus).

Distribuir e modificar tabelas

21/05/2021 • 11 minutes to read

Como distribuir tabelas

Para criar uma tabela distribuída, primeiro você precisa definir o esquema da tabela. Para fazer isso, você pode definir uma tabela usando a instrução [CREATE TABLE](#) da mesma maneira que faria com uma tabela comum do PostgreSQL.

```
CREATE TABLE github_events
(
    event_id bigint,
    event_type text,
    event_public boolean,
    repo_id bigint,
    payload jsonb,
    repo jsonb,
    actor jsonb,
    org jsonb,
    created_at timestamp
);
```

Em seguida, você pode usar a função `create_distributed_table()` para especificar a coluna de distribuição de tabela e criar os fragmentos de trabalho.

```
SELECT create_distributed_table('github_events', 'repo_id');
```

A chamada de função informa à Hiperescala (Citus) de que a tabela `github_events` deve ser distribuída na coluna `repo_id` (realizando hash do valor da coluna). A função também cria fragmentos nos nós de trabalho usando os valores de configuração `citus.shard_count` e `citus.shard_replication_factor`.

Ele cria um número de fragmentos total de `citus.shard_count`, em que cada fragmento é proprietário de uma parte de um espaço de hash e é replicado com base no valor de configuração `citus.shard_replication_factor`. As réplicas de fragmento criadas no trabalho têm as mesmas definições de esquema de tabela, índice e restrição que a tabela no coordenador. Depois que as réplicas são criadas, a função salva todos os metadados distribuídos no coordenador.

Cada fragmento criado recebe uma ID de fragmento exclusiva e todas as réplicas dele têm a mesma ID de fragmento. Os fragmentos são representados no nó de trabalho como tabelas regulares do PostgreSQL nomeadas como '`tablename_shardid`', em que `tablename` é o nome da tabela distribuída e a ID de fragmento é a ID exclusiva atribuída. Você pode se conectar às instâncias de `postgres` de trabalho para exibir ou executar comandos em fragmentos individuais.

Agora você está pronto para inserir dados na tabela distribuída e executar consultas nela. Você também pode obter mais informações sobre o UDF usado nesta seção na referência [DDL de fragmento e tabela](#).

Tabela de referência

O método acima distribui tabelas em vários fragmentos horizontais. Outra possibilidade é distribuir tabelas em um fragmento e replicar o fragmento para cada nó de trabalho. As tabelas distribuídas dessa maneira são chamadas de *tabelas de referência*. Elas são usadas para armazenar dados que precisam ser acessados frequentemente por vários nós em um cluster.

Os candidatos comuns para tabelas de referência incluem:

- Tabelas menores que precisam se unir com tabelas distribuídas maiores.
- Tabelas em aplicativos multilocatário que não têm uma coluna de ID de locatário ou que não estão associadas a um locatário. (Ou, durante a migração, até mesmo algumas tabelas associadas a um locatário.)
- Tabelas que precisam de restrições exclusivas em várias colunas e são pequenas o suficiente.

Por exemplo, suponha que um site de comércio eletrônico multilocatário precise calcular o imposto sobre vendas de transações em qualquer uma das lojas. As informações tributárias não são específicas a nenhum locatário. Faz sentido colocá-las em uma tabela compartilhada. Uma tabela de referência centrada nos EUA pode ser parecida com esta:

```
-- a reference table

CREATE TABLE states (
    code char(2) PRIMARY KEY,
    full_name text NOT NULL,
    general_sales_tax numeric(4,3)
);

-- distribute it to all workers

SELECT create_reference_table('states');
```

Agora, consultas como o cálculo de imposto de um carrinho de compras podem ser unidas com a tabela

`states` sem sobrecarga de rede e podem adicionar uma chave estrangeira ao código de estado para melhor validação.

Além de distribuir uma tabela como um fragmento replicado, o UDF de `create_reference_table` marca-a como uma tabela de referência nas tabelas de metadados de Hiperescala (Citus). A Hiperescala (Citus) executa automaticamente confirmações de duas fases (2PC) para modificações em tabelas marcadas dessa forma, o que fornece garantias de coerência forte.

Se você tiver uma tabela distribuída com uma contagem de fragmentos equivalente a um, poderá atualizá-la para que ela seja uma tabela de referência reconhecida como esta:

```
SELECT upgrade_to_reference_table('table_name');
```

Para obter outro exemplo de como usar tabelas de referência, confira o [tutorial de banco de dados de multilocatários](#).

Como distribuir dados do coordenador

Se um banco de dados do PostgreSQL existente for convertido no nó coordenador de um cluster de Hiperescala (Citus), os dados nas tabelas poderão ser distribuídos para um aplicativo com eficiência e interrupção mínima.

A função `create_distributed_table` descrita anteriormente funciona em tabelas vazias e não vazias e, para as últimas, ela distribui automaticamente linhas de tabela em todo o cluster. Você saberá se ela copiar dados pela presença da mensagem "NOTICE: Copying data from local table...", por exemplo:

```
CREATE TABLE series AS SELECT i FROM generate_series(1,1000000) i;
SELECT create_distributed_table('series', 'i');
NOTICE:  Copying data from local table...
create_distributed_table
-----
(1 row)
```

As gravações na tabela são bloqueadas enquanto os dados são migrados e as gravações pendentes são tratadas como consultas distribuídas quando a função é confirmada. (Se a função falhar, as consultas se tornarão locais novamente.) As leituras podem continuar normalmente e se tornarão consultas distribuídas quando a função for confirmada.

Ao distribuir as tabelas A e B, em que um tem uma chave estrangeira para B, distribua primeiro a tabela de destino da chave B. Fazer isso na ordem errada causará um erro:

```
ERROR: cannot create foreign key constraint
DETAIL: Referenced table must be a distributed table or a reference table.
```

Se não for possível distribuir na ordem correta, remova as chaves estrangeiras, distribua as tabelas e recrie as chaves estrangeiras.

Ao migrar dados de um banco de dados externo, como de um Amazon RDS para a Nuvem de Hiperescala (Citus), primeiro crie as tabelas distribuídas de Hiperescala (Citus) por meio do `create_distributed_table` e depois copie os dados para a tabela. A cópia em tabelas distribuídas evita que o nó coordenador fique sem espaço.

Como colocar tabelas

A colocação significa manter informações relacionadas nos mesmos computadores. Ela permite consultas eficientes, ao mesmo tempo que aproveita a escalabilidade horizontal de todo o conjunto de dados. Para obter mais informações, confira [colocação](#).

As tabelas são colocadas em grupos. Para controlar manualmente a atribuição de grupo de colocação de uma tabela, use o parâmetro `colocate_with` opcional de `create_distributed_table`. Se você não se importa com uma colocação de tabela, omita esse parâmetro. Ele usa como padrão o valor `'default'`, que agrupa a tabela com qualquer outra tabela de colocação padrão que tenha o mesmo tipo de coluna de distribuição, contagem de fragmentos e fator de replicação. Se você quiser interromper ou atualizar essa colocação implícita, poderá usar `update_distributed_table_colocation()`.

```
-- these tables are implicitly co-located by using the same
-- distribution column type and shard count with the default
-- co-location group

SELECT create_distributed_table('A', 'some_int_col');
SELECT create_distributed_table('B', 'other_int_col');
```

Quando uma nova tabela não estiver relacionada a outras no grupo de colocações implícitas hipotéticas, especifique `colocated_with => 'none'`.

```
-- not co-located with other tables

SELECT create_distributed_table('A', 'foo', colocate_with => 'none');
```

Dividir tabelas não relacionadas nos próprios grupos de colocação aprimora o desempenho do [rebalanceamento de fragmentos](#), pois os fragmentos no mesmo grupo precisam ser movidos juntos.

Quando as tabelas estão de fato relacionadas (por exemplo, quando elas forem unidas), faz sentido colocá-las explicitamente. Os ganhos da colocação apropriada são mais importantes do que qualquer sobrecarga de rebalanceamento.

Para colocar explicitamente várias tabelas, distribua uma e coloque as outras no grupo de colocação. Por exemplo:

```
-- distribute stores
SELECT create_distributed_table('stores', 'store_id');

-- add to the same group as stores
SELECT create_distributed_table('orders', 'store_id', colocate_with => 'stores');
SELECT create_distributed_table('products', 'store_id', colocate_with => 'stores');
```

As informações sobre grupos de colocação são armazenadas na tabela [pg_dist_colocation](#), enquanto [pg_dist_partition](#) revela quais tabelas são atribuídas a quais grupos.

Como remover tabelas

Você pode usar o comando `DROP TABLE` padrão do PostgreSQL para remover as suas tabelas distribuídas. Assim como acontece com tabelas regulares, o `DROP TABLE` remove todos os índices, regras, gatilhos e restrições existentes na tabela de destino. Além disso, ele também remove os fragmentos nos nós de trabalho e limpa os metadados.

```
DROP TABLE github_events;
```

Como modificar tabelas

A Hiperescala (Citus) propaga automaticamente muitos tipos de instruções DDL. A modificação de uma tabela distribuída no nó coordenador também atualizará os fragmentos nos trabalhos. Outras instruções DDL exigem a propagação manual e algumas outras são proibidas, como as que modificariam uma coluna de distribuição. A tentativa de executar a DDL que não é elegível para a propagação automática gera um erro e deixa as tabelas no nó coordenador inalteradas.

Veja abaixo uma referência das categorias de instruções DDL que se propagam. A propagação automática pode ser habilitada ou desabilitada com um [parâmetro de configuração](#)

Como adicionar/modificar colunas

A Hiperescala (Citus) propaga a maioria dos comandos `ALTER TABLE` automaticamente. Adicionar colunas ou alterar os valores padrão delas funciona da mesma forma que em um banco de dados PostgreSQL de computador único:

```
-- Adding a column

ALTER TABLE products ADD COLUMN description text;

-- Changing default value

ALTER TABLE products ALTER COLUMN price SET DEFAULT 7.77;
```

Alterações significativas em uma coluna existente, como renomeá-la ou alterar o tipo de dados dela, também podem ser usadas. No entanto, o tipo de dados da [coluna de distribuição](#) não pode ser alterado. Essa coluna determina como os dados de tabela são distribuídos por meio do cluster de Hiperescala (Citus) e modificar o tipo de dados exigiria a movimentação dos dados.

Tentar fazer isso causa um erro:

```
-- assuming store_id is the distribution column
-- for products, and that it has type integer

ALTER TABLE products
ALTER COLUMN store_id TYPE text;

/*
ERROR:  XX000: cannot execute ALTER TABLE command involving partition column
LOCATION:  ErrorIfUnsupportedAlterTableStmt, multi_utility.c:2150
*/
```

Como adicionar/remover restrições

Usar a Hiperescala (Citus) permite que você continue a desfrutar da segurança de um banco de dados relacional, incluindo as restrições de banco de dados (confira as [documentações](#) do PostgreSQL). Devido à natureza dos sistemas distribuídos, a Hiperescala (Citus) não faz referência cruzada das restrições de exclusividade ou da integridade referencial entre nós de trabalho.

Para configurar uma chave estrangeira entre tabelas distribuídas colocadas, inclua sempre a coluna de distribuição na chave. A inclusão da coluna de distribuição pode envolver a criação do composto de chaves.

As chaves estrangeiras podem ser criadas nestas situações:

- entre duas tabelas locais (não distribuídas),
- entre duas tabelas de referência,
- entre duas tabelas distribuídas [colocadas](#) quando a chave inclui a coluna de distribuição ou
- como uma tabela distribuída que faz referência a uma [tabela de referência](#)

Não há suporte para chaves estrangeiras de tabelas de referência para tabelas distribuídas.

NOTE

As chaves primárias e as restrições de exclusividade precisam incluir a coluna de distribuição. Adicioná-las a uma coluna de não distribuição gera um erro

Este exemplo mostra como criar chaves primárias e estrangeiras em tabelas distribuídas:

```

-- 
-- Adding a primary key
-- -----
-- We'll distribute these tables on the account_id. The ads and clicks
-- tables must use compound keys that include account_id.

ALTER TABLE accounts ADD PRIMARY KEY (id);
ALTER TABLE ads ADD PRIMARY KEY (account_id, id);
ALTER TABLE clicks ADD PRIMARY KEY (account_id, id);

-- Next distribute the tables

SELECT create_distributed_table('accounts', 'id');
SELECT create_distributed_table('ads',      'account_id');
SELECT create_distributed_table('clicks',   'account_id');

-- 
-- Adding foreign keys
-- -----
-- Note that this can happen before or after distribution, as long as
-- there exists a uniqueness constraint on the target column(s) which
-- can only be enforced before distribution.

ALTER TABLE ads ADD CONSTRAINT ads_account_fk
    FOREIGN KEY (account_id) REFERENCES accounts (id);
ALTER TABLE clicks ADD CONSTRAINT clicks_ad_fk
    FOREIGN KEY (account_id, ad_id) REFERENCES ads (account_id, id);

```

Da mesma forma, inclua a coluna de distribuição nas restrições de exclusividade:

```

-- Suppose we want every ad to use a unique image. Notice we can
-- enforce it only per account when we distribute by account id.

ALTER TABLE ads ADD CONSTRAINT ads_unique_image
    UNIQUE (account_id, image_url);

```

As restrições não nulas podem ser aplicadas a qualquer coluna (de distribuição ou não) porque elas não exigem pesquisas entre os trabalhos.

```
ALTER TABLE ads ALTER COLUMN image_url SET NOT NULL;
```

Como usar restrições NOT VALID

Em algumas situações, pode ser útil impor restrições em novas linhas e, ao mesmo tempo, permitir que linhas sem conformidade existentes permaneçam inalteradas. A Hiperescala (Citus) dá suporte a esse recurso para restrições CHECK e chaves estrangeiras usando a designação de restrição "NOT VALID" do PostgreSQL.

Por exemplo, considere um aplicativo que armazena perfis de usuário em uma [tabela de referência](#).

```

-- we're using the "text" column type here, but a real application
-- might use "citext" which is available in a postgres contrib module

CREATE TABLE users ( email text PRIMARY KEY );
SELECT create_reference_table('users');

```

No decorrer do tempo, imagine que alguns endereços são inseridos na tabela.

```
INSERT INTO users VALUES
  ('foo@example.com'), ('hacker12@aol.com'), ('lol');
```

Gostaríamos de validar os endereços, mas o PostgreSQL normalmente não nos permite adicionar uma restrição CHECK com falha nas linhas existentes. No entanto, ele *permite* uma restrição marcada como inválida:

```
ALTER TABLE users
ADD CONSTRAINT syntactic_email
CHECK (email ~
'^[a-zA-Z0-9.!#$%&'*+/=?^`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9]))?*$'
) NOT VALID;
```

Novas linhas agora estão protegidas.

```
INSERT INTO users VALUES ('fake');

/*
ERROR: new row for relation "users_102010" violates
       check constraint "syntactic_email_102010"
DETAIL: Failing row contains (fake).
*/
```

Mais tarde, fora do horário de pico, um administrador de banco de dados pode tentar consertar as linhas inválidas e revalidar a restrição.

```
-- later, attempt to validate all rows
ALTER TABLE users
VALIDATE CONSTRAINT syntactic_email;
```

A documentação do PostgreSQL tem mais informações sobre NOT VALID e VALIDATE CONSTRAINT na seção [ALTER TABLE](#).

Como adicionar/remover índices

A Hiperescala (Citus) dá suporte à adição e remoção de [índices](#):

```
-- Adding an index

CREATE INDEX clicked_at_idx ON clicks USING BRIN (clicked_at);

-- Removing an index

DROP INDEX clicked_at_idx;
```

A adição de um índice usa um bloqueio de gravação, que pode ser indesejável em um "sistema de registro" de multilocalatório. Para minimizar o tempo de inatividade do aplicativo, crie o índice [simultaneamente](#). Esse método requer mais trabalho total do que um build de índice padrão e leva mais tempo para ser concluído. No entanto, como ele permite que as operações normais continuem enquanto o índice é criado, esse método é útil para adicionar novos índices em um ambiente de produção.

```
-- Adding an index without locking table writes

CREATE INDEX CONCURRENTLY clicked_at_idx ON clicks USING BRIN (clicked_at);
```

Configurar alta disponibilidade de Hiperescala (Citus)

09/08/2021 • 2 minutes to read

O Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus) fornece alta disponibilidade (HA) para evitar o tempo de inatividade do banco de dados. Com a HA habilitada, cada nó em um grupo de servidores recebe um modo de espera. Se o nó original não estiver íntegro, seu modo de espera será promovido para substituí-lo.

IMPORTANT

Como a HA duplica o número de servidores no grupo, ela também dobra o custo.

É possível habilitar a HA durante a criação do grupo de servidores ou posteriormente na guia **Computação + armazenamento** para seu grupo de servidores no portal do Azure. A interface do usuário é semelhante em ambos os casos. Arraste o controle deslizante de **Alta disponibilidade** de NÃO para SIM:

Search (Cmd+ /) <

Overview

Activity log

Tags

Compute + storage

Connection strings

Roles

Coordinator node parameters

Worker node parameters

Locks

Networking

New support request

You can scale your Hyperscale (Citus) cluster by adding nodes to it without cluster downtime and by scaling compute cores on coordinator and worker nodes. [Learn more](#)

Worker nodes

Expand your server group and scale your database by adding worker nodes. Select up to 64 vCores with 8 GiB RAM per vCore and up to 2 TiB of storage with up to 3 IOPS / GiB per node.

Worker node count: 2 nodes
If you need more than 20 nodes, [contact us](#)

Configuration (per worker node):

- vCores: 4 vCores
- Storage: 0.5 TiB

Your configuration can use up to 3 IOPS / GiB.

Coordinator node

The coordinator node coordinates your queries and manages your schema. Configure your coordinator node performance by selecting CPU vCore and storage capacity. Select up to 64 vCores with 4 GiB RAM per vCore and up to 2 TiB of storage with up to 3 IOPS / GiB per node.

Configuration (coordinator node):

- vCores: 4 vCores
- Storage: 0.5 TiB

Your configuration can use up to 3 IOPS / GiB.

High availability

Enable high availability to have standby replicas of every node with automatic failover in your server group.

Enable high availability: Yes

Save

Clique no botão **Salvar** para aplicar sua seleção. Habilitar a HA pode levar algum tempo, pois o grupo de servidores provisiona esperas e transmite dados para elas.

A guia **Visão geral** do grupo de servidores listará todos os nós e suas esperas, além de uma coluna **Alta disponibilidade** que indica se ela foi habilitada com sucesso para cada nó.

Type	Status	High availability
Coordinator	Available	 Healthy
Worker	Available	 Healthy
Worker	Available	 Healthy

Próximas etapas

Saiba mais sobre a [alta disponibilidade](#).

Criar e gerenciar réplicas de leitura no Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus) do portal do Azure

21/05/2021 • 2 minutes to read

IMPORTANT

As réplicas de leitura no Hiperescala (Citus) estão atualmente em versão prévia. Essa versão prévia é fornecida sem um contrato de nível de serviço e não é recomendada para cargas de trabalho de produção. Alguns recursos podem não ter suporte ou podem ter restrição de recursos.

É possível ver a lista completa dos novos recursos em [Versão prévia dos recursos para Hiperescala \(Citus\)](#).

Neste artigo, você aprenderá a criar e gerenciar réplicas de leitura no Hiperescala (Citus) no portal do Azure. Para saber mais sobre réplicas de leitura, confira [Visão Geral](#).

Pré-requisitos

Um [grupo de servidores de Hiperescala \(Citus\)](#) para ser o primário.

Criar uma réplica de leitura

Para criar uma réplica de leitura, siga estas etapas:

1. Selecione um grupo de servidores do Banco de Dados do Azure para PostgreSQL existente a ser usado como primário.
2. Na barra lateral grupo de servidores, em **Gerenciamento de grupos de servidores**, selecione **Replicação**.
3. Selecione **para adicionar réplica**.
4. Insira um nome para a réplica de leitura.
5. Selecione **OK** para confirmar a criação da réplica.

Depois que a réplica de leitura for criada, ela poderá ser exibida na janela **Replicação**.

IMPORTANT

Examine a [seção considerações da Visão Geral da Réplica de leitura](#).

Antes que uma configuração do grupo de servidores primário seja atualizada para um novo valor, atualize a configuração de réplica para um valor igual ou maior. Esta ação ajuda a réplica a acompanhar as alterações feitas ao mestre.

Excluir um grupo de servidores primário

Para excluir um grupo de servidores primário, realize as mesmas etapas para excluir um grupo de servidores Hiperescala (Citus) autônomo.

IMPORTANT

Ao excluir um grupo de servidores primário, a replicação para todas as réplicas de leitura será interrompida. As réplicas de leitura tornam-se grupos de servidores autônomos que agora têm suporte para leitura e gravação.

Para excluir um grupo de servidores do portal do Azure, siga estas etapas:

1. No portal do Azure, selecione o grupo de servidores primário do Banco de Dados do Azure para PostgreSQL.
2. Abra a página **Visão geral** para o grupo de servidores. Selecione **Excluir**.
3. Insira o nome do grupo de servidores primário a ser excluído. Selecione **Excluir** para confirmar a exclusão do grupo de servidores primário.

Excluir uma réplica

É possível excluir uma réplica de leitura da mesma forma como exclui um grupo de servidores primário.

- No portal do Azure, abra a página **Visão geral** para a réplica de leitura. Selecione **Excluir**.

Você também pode excluir a réplica de leitura usando a janela **Replicação** seguindo estas etapas:

1. No portal do Azure, selecione seu grupo de servidores primário Hiperescala (Citus).
2. No menu de grupo de servidores, em **Gerenciamento de grupos de servidores**, selecione **Replicação**.
3. Selecione a réplica de leitura a excluir.
4. Selecione **Excluir réplica**.
5. Insira o nome da réplica a excluir. Selecione **Excluir** para confirmar a exclusão da réplica.

Próximas etapas

- Saiba mais sobre [Réplicas de leitura no Banco de Dados do Azure para PostgreSQL – Hiperescala \(Citus\)](#).

Gerenciar configurações de manutenção agendada para o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 2 minutes to read

É possível especificar opções de manutenção para cada grupo de servidores de Hiperescala (Citus) na assinatura do Azure. As opções incluem o agendamento de manutenção e as configurações de notificação para eventos de manutenção futuros e concluídos.

Pré-requisitos

Para concluir este guia de instruções, você precisa:

- Um [grupo de servidores Hiperescala \(Citus\) – Banco de Dados do Azure para PostgreSQL](#)

Especificar opções de agendamento de manutenção

1. Na página grupo de servidores do Hiperescala (Citus), no título **Configurações**, escolha **Manutenção** para abrir as opções de manutenção agendada.
2. O agendamento padrão (gerenciado pelo sistema) é um dia da semana aleatório e uma janela de 30 minutos para a manutenção começar entre 23h e 7h do horário do grupo de servidores da [região do Azure](#). Se você quiser personalizar esse agendamento, escolha **Personalizar agendamento**. Em seguida, é possível selecionar um dia da semana de preferência e uma janela de 30 minutos para a hora de início da manutenção.

Notificações sobre eventos de manutenção agendada

É possível usar a Integridade do Serviço do Azure para [exibir notificações](#) sobre a manutenção agendada futura e passada no grupo de servidores de Hiperescala (Citus). Também é possível [Configurar](#) alertas na Integridade do Serviço do Azure para obter notificações sobre eventos de manutenção.

Próximas etapas

- Saiba mais sobre [manutenção agendada no Banco de Dados do Azure para PostgreSQL – Hiperescala \(Citus\)](#)
- Saiba mais sobre a [Integridade do Serviço do Azure](#)

Use o portal do Azure para configurar alertas sobre métricas para o Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

21/05/2021 • 3 minutes to read

Este artigo mostra como configurar alertas do Banco de Dados do Azure para PostgreSQL usando o Portal do Azure. É possível receber um alerta com base em [métricas de monitoramento](#) dos seus serviços do Azure.

Vamos configurar o disparo de um alerta quando o valor de uma métrica especificada ultrapassar um limite. O alerta é disparado quando a condição é atendida pela primeira vez e continua sendo disparado posteriormente.

Você pode configurar um alerta para fazer as seguintes ações quando ele disparar:

- Enviar notificações por email ao administrador e aos coadministradores do serviço.
- Enviar um email para outros emails que você especificar.
- Chamar um webhook.

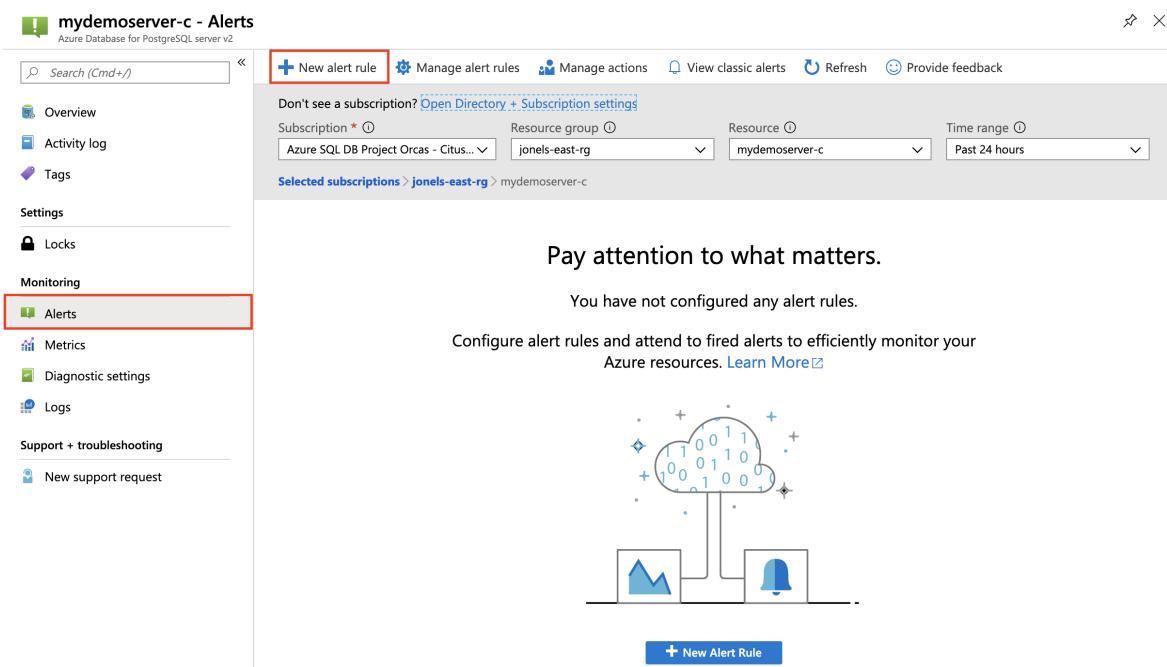
Você pode configurar e obter informações sobre as regras de alerta usando:

- [Azure portal](#)
- [CLI do Azure](#)
- [API REST do Azure Monitor](#)

Criar uma regra de alerta em uma métrica no Portal do Azure

1. No [Portal do Azure](#), selecione o servidor do Banco de Dados do Azure para PostgreSQL que você deseja monitorar.

2. Na seção **Monitoramento** da barra lateral, selecione **Alertas** como mostrado abaixo:



3. Clique em **Nova regra de alerta** (ícone +).

4. A página **Criar regra** é aberta, conforme mostrado abaixo. Preencha as informações obrigatórias:

The screenshot shows the 'Create rule' page under 'Rules management'. The top navigation bar includes 'Home > mydemoserver-c - Alerts > Create rule'. The main area is titled 'Create rule' with a 'Select' button. The 'RESOURCE' section shows 'mydemoserver-c' selected. The 'HIERARCHY' section shows 'Azure SQL DB Project Orcas - CitusData' under 'jonels-east-rg'. The 'CONDITION' section has a note: 'No condition defined, click on 'Add condition' to select a signal and define its logic' with a red box around the 'Add' button. The 'ACTIONS' section shows 'Action group name' with 'No action group selected' and buttons for 'Select action group' and 'Create action group'. A note at the bottom says: 'Action rules (preview) allows you to define actions at scale as well as suppress actions. Learn more about this functionality [here](#)'. The 'ALERT DETAILS' section includes 'Alert rule name' (with a red box around the input field), 'Description' (with a red box around the input field), and 'Enable rule upon creation' with 'Yes' selected.

5. Na seção **Condição**, selecione **Adicionar**.

6. Selecione uma métrica da lista de sinais sobre a qual deseja ser alertado. Neste exemplo, selecione "Porcentagem de armazenamento".

The screenshot shows the 'Configure signal logic' dialog. It has two dropdown menus: 'Signal type' (set to 'All') and 'Monitor service' (set to 'All'). Below them is a note: 'Displaying 1 - 12 signals out of total 12 signals'. A search bar 'Search by signal name' is followed by a table with 12 rows, each representing a signal. The table has columns: 'Signal name', 'Signal type', and 'Monitor service'. The first 11 rows have 'Signal type' set to 'Metric' and 'Monitor service' set to 'Platform'. The last row has 'Signal type' set to 'Activity Log' and 'Monitor service' set to 'Administrative'. A red box highlights the entire table. At the bottom is a 'Done' button.

Signal name	Signal type	Monitor service
CPU percent	Metric	Platform
Memory percent	Metric	Platform
IOPS	Metric	Platform
Storage percent	Metric	Platform
Storage used	Metric	Platform
Active Connections	Metric	Platform
Network Out	Metric	Platform
Network In	Metric	Platform
All Administrative operations	Activity Log	Administrative
Create/Update PostgreSQL Server (Microsoft.DBforPostgreSQL/serversv2)	Activity Log	Administrative
Delete PostgreSQL Server (Microsoft.DBforPostgreSQL/serversv2)	Activity Log	Administrative
Batch Update Server Configurations (Microsoft.DBforPostgreSQL/serversv2)	Activity Log	Administrative

7. Configure a lógica de alerta:

- **Operador** (por exemplo, "Maior que")
- **Valor do limite** (por exemplo, 85%)
- **Granularidade de agregação**, o período durante o qual a regra de métrica deverá ser atendida antes de o alerta disparar (por exemplo, "Os últimos 30 minutos")
- e **Frequência de avaliação** (por exemplo, "1 minuto")

Selecione Concluído ao concluir.

Alert logic

Threshold ⓘ

Operator ⓘ	Aggregation type * ⓘ	Threshold value * ⓘ
Greater than	Average	85 %

Condition preview

Whenever the storage percent is greater than 85 percent

Evaluated based on

Aggregation granularity (Period) * ⓘ	Frequency of evaluation ⓘ
30 minutes	Every 1 Minute

Done

8. Dentro da seção **Grupos de Ações**, selecione **Criar Novo** para criar um novo grupo para receber notificações sobre o alerta.

9. Preencha o formulário "Adicionar grupo de ações" com um nome, o nome curto, a assinatura e o grupo de recursos.

Home > mydemoserver-c - Alerts > Create rule > Add action group

Add action group

Action group name * ⓘ	newactiongroup
Short name * ⓘ	group1
Subscription * ⓘ	Azure SQL DB Project Orcas - CitusData
Resource group * ⓘ	Default-ActivityLogAlerts

Actions

Action Name *	Action Type *	Status	Configure	Actions
newactiongroup	Email/SMS/Push/Voice		Edit details	X

Please configure the action by clicking the link.

Unique name for the action Select an action type

[Privacy Statement](#)

[Pricing](#)

! Have a consistent format in emails, notifications and other endpoints irrespective of monitoring source. You can enable per action by editing details.
[Learn more](#)

10. Configure o tipo de ação Email/SMS/Push/Voz.

Escolha "Enviar email para a Função do Azure Resource Manager" para enviar notificações aos proprietários da assinatura, colaboradores e leitores.

Selecione OK ao concluir.

The screenshot shows two side-by-side configuration pages. On the left, the 'Add action group' page has fields for Action group name (newactiongroup), Short name (group1), Subscription (Azure SQL DB Project Orcas - CitusData), and Resource group (Default-ActivityLogAlerts). Under Actions, there's a note to 'Please configure the action by clicking the link.' Below this are sections for Privacy Statement and Pricing. A tooltip suggests having a consistent format for emails and notifications. On the right, the 'Email/SMS/Push/Voice' page is shown with the 'Email' option selected. It includes fields for Email (email@contoso.com), SMS (unchecked), Phone number (1234567890), Azure app Push Notifications (unchecked), Voice (unchecked), and a section to enable the common alert schema with options 'Yes' or 'No'. A large 'OK' button is at the bottom.

11. Especifique um Nome da regra de alerta, uma Descrição e uma Gravidade.

The screenshot shows the 'ALERT DETAILS' configuration page. It includes fields for Alert rule name (Storage percentage greater tha 85), Description (Storage percentage greater tha 85), Severity (Sev 3), and an 'Enable rule upon creation' switch set to 'Yes'. A tooltip indicates it may take up to 10 minutes for the metric alert rule to become active.

12. Selecione Criar regra de alerta para criar o alerta.

Em alguns minutos, o alerta estará ativo e disparará conforme descrito anteriormente.

Gerenciando alertas

Depois de criar um alerta, é possível selecioná-lo e executar as seguintes ações:

- Exibir um gráfico mostrando o limite de métrica e os valores reais do dia anterior relevante para este alerta.
- Editar ou Excluir a regra de alerta.

- Desabilitar ou Habilitar o alerta, se desejar interromper temporariamente ou retomar o recebimento de notificações.

Alertas sugeridos

Espaço em disco

O monitoramento e o alerta são importantes para cada grupo de servidores Hiperescala (Citus) de produção. O banco de dados PostgreSQL subjacente requer espaço livre em disco para funcionar corretamente. Se o disco ficar cheio, o nó de servidor de banco de dados ficará offline e se recusará a iniciar até que o espaço esteja disponível. Nesse ponto, é necessária uma solicitação de suporte da Microsoft para corrigir a situação.

É recomendável definir alertas de espaço em disco em cada nó de cada grupo de servidores, mesmo para uso de não produção. Os alertas de uso de espaço em disco enviam o aviso antecipado necessário para intervir e manter os nós íntegros. Para obter melhores resultados, experimente uma série de alertas em 75%, 85% e 95% de uso. Os percentuais a serem escolhidos dependem da velocidade de ingestão de dados. Quando ela é rápida, o disco enche de modo mais acelerado.

À medida que o disco se aproximar do limite de espaço, tente essas técnicas para obter mais espaço livre:

- Examinar a política de retenção de dados. Mova os dados mais antigos para um armazenamento frio, se possível.
- Considere [adicionar nós](#) ao grupo de servidores e rebalancear os fragmentos. O rebalanceamento distribui os dados em mais computadores.
- Considere [aumentar a capacidade](#) dos nós de trabalho. Cada trabalho pode ter até 2 TiB de armazenamento. No entanto, tente adicionar nós antes de redimensioná-los, porque a adição de nós é concluída mais rapidamente.

Uso da CPU

O monitoramento do uso da CPU é útil para estabelecer uma linha de base de desempenho. Por exemplo, talvez você observe que o uso da CPU geralmente está em cerca de 40% a 60%. Se o uso da CPU começar a ficar em cerca de 95%, será uma anomalia. O uso da CPU pode refletir o crescimento orgânico, mas também pode revelar uma consulta isolada. Ao criar um alerta de CPU, defina uma granularidade de agregação longa para detectar aumentos prolongados e ignorar picos momentâneos.

Próximas etapas

- Saiba mais sobre como [configurar webhooks em alertas](#).
- Tenha uma [visão geral da coleção de métricas](#) para verificar se o serviço está disponível e responsivo.

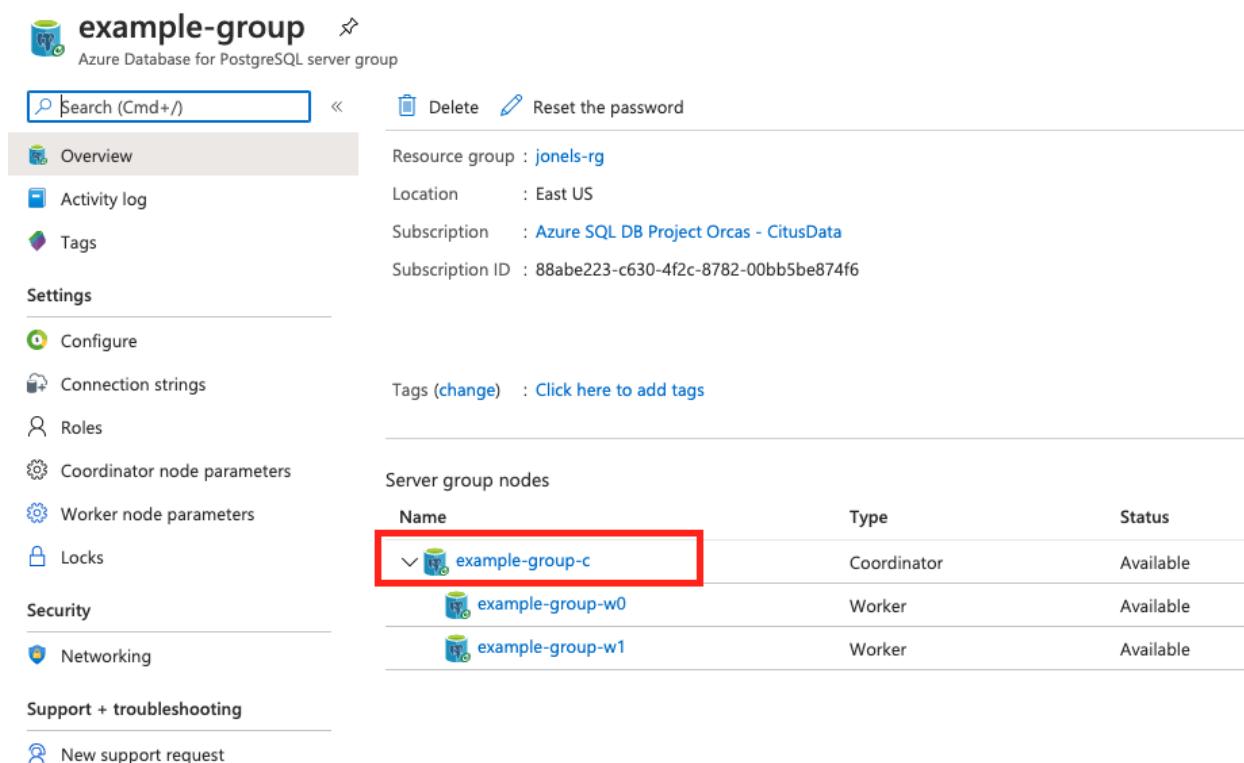
Logs no banco de dados do Azure para PostgreSQL-Citus (hiperescala)

21/05/2021 • 2 minutes to read

Os logs do PostgreSQL estão disponíveis em todos os nós de um grupo de servidores de hiperescala (Citus). Você pode enviar logs para um servidor de armazenamento ou para um serviço de análise. Os logs podem ser usados para identificar, solucionar problemas e reparar erros de configuração e desempenho inferior.

Acessando os logs

Para acessar os logs do PostgreSQL para um coordenador ou nó de trabalho de hiperescala (Citus), abra o nó no portal do Azure:



The screenshot shows the Azure portal interface for managing a PostgreSQL server group named 'example-group'. The left sidebar contains navigation links for Overview, Activity log, Tags, Settings (Configure, Connection strings, Roles, Coordinator node parameters, Worker node parameters, Locks), Security, Networking, Support + troubleshooting (New support request), and Help + feedback. The main content area displays the 'Overview' tab information: Resource group: 'jonels-rg', Location: 'East US', Subscription: 'Azure SQL DB Project Orcas - CitusData', and Subscription ID: '88abe223-c630-4f2c-8782-00bb5be874f6'. Below this, there is a 'Tags (change)' section with a link to 'Click here to add tags'. The 'Server group nodes' section lists three nodes: 'example-group-c' (Coordinator, Available), 'example-group-w0' (Worker, Available), and 'example-group-w1' (Worker, Available). The 'example-group-c' node is highlighted with a red box.

Para o nó selecionado, abra configurações de diagnóstico e clique em + Adicionar configuração de diagnóstico.

example-group-c | Diagnostic settings ⚡

Azure Database for PostgreSQL server v2

Search (Cmd+/) Refresh Provide feedback

Overview Activity log Tags

Settings

Locks

Monitoring

Alerts Metrics **Diagnostic settings** 1

Logs

Support + troubleshooting

New support request

Diagnostic settings are used to configure streaming export of platform logs and metrics for a resource to the [diagnostics settings](#)

Diagnostics settings

Name	Storage account
No diagnostic settings defined	

[+ Add diagnostic setting](#) 2

Click 'Add Diagnostic setting' above to configure the collection of the following data:

- PostgreSQLLogs
- AllMetrics

Escolha um nome para as novas configurações de diagnóstico e marque a caixa PostgreSQLLogs . Escolha quais destinos devem receber os logs.

Home > example-group-c | Diagnostic settings >

Diagnostics setting

Save Discard Delete Provide feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name *

example-group-c-logs 1

Category details

log

PostgreSQLLogs 2

metric

AllMetrics

Destination details

Send to Log Analytics 3

Archive to a storage account

Stream to an event hub

Próximas etapas

- [Introdução às consultas do log Analytics](#)
- [Saiba mais sobre os hubs de eventos do Azure](#)

Restauração pontual de um grupo de servidores de Hiperescala (Citus)

21/05/2021 • 2 minutes to read

Este artigo fornece procedimentos passo a passo para realizar [recuperações pontuais](#) para um grupo de servidores de Hiperescala (Citus) usando backups. Você pode restaurar para o backup mais antigo ou para um ponto de restauração personalizado dentro do período de retenção.

Como restaurar para o ponto de restauração mais antigo

Siga estas etapas para restaurar o grupo de servidores de Hiperescala (Citus) para o backup mais antigo existente.

1. No [portal do Azure](#), escolha o grupo de servidores que você deseja restaurar.
2. Clique em **Visão geral** no painel esquerdo e clique em **Restaurar**.

IMPORTANT

Se o botão **Restaurar** ainda não estiver disponível para o grupo de servidores, abra uma solicitação de suporte do Azure.

3. A página de restauração solicitará que você escolha entre o ponto de restauração **mais antigo** e o **personalizado** e exibirá a data mais antiga.
4. Selecione **Ponto de restauração mais antigo**.
5. Forneça um novo nome de grupo de servidores no campo **Restaurar para o novo servidor**. Os outros campos (assinatura, grupo de recursos e local) são exibidos, mas não podem ser editados.
6. Clique em **OK**.
7. Será mostrada uma notificação de que a operação de restauração foi iniciada.

Por fim, siga as [tarefas pós-restauração](#).

Como restaurar para um ponto de restauração personalizado

Siga estas etapas para restaurar o grupo de servidores de Hiperescala (Citus) para uma data e hora de sua escolha.

1. No [portal do Azure](#), escolha o grupo de servidores que você deseja restaurar.
2. Clique em **Visão geral** no painel esquerdo e clique em **Restaurar**

IMPORTANT

Se o botão **Restaurar** ainda não estiver disponível para o grupo de servidores, abra uma solicitação de suporte do Azure.

3. A página de restauração solicitará que você escolha entre o ponto de restauração **mais antigo** e o **personalizado** e exibirá a data mais antiga.

4. Escolha **Ponto de restauração personalizado**.
5. Selecione data e hora para o **Ponto de restauração (UTC)** e forneça um novo nome do grupo de servidores no campo **Restaurar para o novo servidor**. Os outros campos (assinatura, grupo de recursos e local) são exibidos, mas não podem ser editados.
6. Clique em **OK**.
7. Será mostrada uma notificação de que a operação de restauração foi iniciada.

Por fim, siga as [tarefas pós-restauração](#).

Tarefas de pós-restauração

Depois de uma restauração, você deve fazer o seguinte para que os usuários e aplicativos voltem a funcionar:

- Se o novo servidor é usado para substituir o servidor original, redirecione clientes e aplicativos de cliente para o novo servidor
- Verifique se há um firewall adequado no nível do servidor para que os usuários se conectem. Essas regras não são copiadas no grupo de servidores original.
- Ajuste os parâmetros do servidor PostgreSQL conforme necessário. Os parâmetros não são copiados no grupo de servidores original.
- Verifique se as permissões e os logons adequados no nível do banco de dados estão em vigor.
- Configure os alertas conforme apropriado.

Próximas etapas

- Saiba mais sobre [backup e restauração](#) em Hiperescala (Citus).
- Defina os [alertas sugeridos](#) nos grupos de servidores de Hiperescala (Citus).

Criar um grupo de servidores de Hiperescala (Citus)

21/05/2021 • 2 minutes to read

Estas instruções descrevem como atualizar para uma nova versão principal do PostgreSQL em todos os nós do grupo de servidores.

Testar a atualização primeiro

Atualizar o PostgreSQL causa mais alterações do que você imagina, pois a Hiperescala (Citus) também atualizará as [extensões de banco de dados](#), incluindo a extensão Citus. É altamente recomendável que você teste seu aplicativo com a nova versão do PostgreSQL e do Citus antes de atualizar o ambiente de produção.

Uma forma conveniente de testar é fazer uma cópia do grupo de servidores usando a [restauração pontual](#). Atualize a cópia e teste seu aplicativo em relação a ela. Depois de verificar que tudo funciona corretamente, atualize o grupo de servidores original.

Atualizar um grupo de servidores no portal do Azure

1. Na seção **Visão geral** de um grupo de servidores de Hiperescala (Citus), selecione o botão **Atualizar**.
2. Uma caixa de diálogo é exibida, mostrando a versão atual do PostgreSQL e do Citus. Escolha uma nova versão do PostgreSQL na lista **Atualizar para**.
3. Verifique se o valor em **Versão do Citus após a atualização** é o que você espera. Esse valor é alterado com base na versão do PostgreSQL que você seleciona.
4. Selecione o botão **Atualizar** para continuar.

Próximas etapas

- Saiba mais sobre [Versões compatíveis do PostgreSQL](#).
- Veja [quais extensões](#) são empacotadas com cada versão do PostgreSQL em um grupo de servidores de Hiperescala (Citus).

Solucionar problemas de conexão ao Banco de Dados do Azure para PostgreSQL - Hiperescala (Citus)

21/05/2021 • 3 minutes to read

Problemas de conexão podem ter várias causas, como:

- Configurações de firewall
- Tempo limite da conexão
- Informações de entrada incorretas
- Limite de conexão atingido para o grupo de servidores
- Problemas com a infraestrutura do serviço
- Manutenção de serviço
- O nó do coordenador que faz failover para um novo hardware

Normalmente, problemas de conexão com Hiperescala (Citus) podem ser classificados da seguinte forma:

- Erros transitórios (de curta duração ou intermitentes)
- Erros persistentes ou não transitórios (erros regularmente recorrentes)

Solucionar problemas de erros transitórios

Erros transitórios ocorrem por vários motivos. Os mais comuns incluem manutenção do sistema, erro com hardware ou software e atualizações de vCore do nó coordenador.

Habilitar a alta disponibilidade para nós do grupo de servidores de Hiperescala (Citus) pode mitigar esses tipos de problemas automaticamente. No entanto, seu aplicativo ainda deve estar preparado para perder a conexão brevemente. Além disso, outros eventos podem levar mais tempo para serem mitigados, como quando uma transação grande causa uma recuperação de execução longa.

Etapas para resolver problemas de conectividade temporários

1. Confira o [Painel de Serviços do Microsoft Azure](#) quanto a quaisquer interrupções conhecidas que tenham ocorrido durante o tempo em que o aplicativo estava relatando erros.
2. Os aplicativos que se conectam a um serviço de nuvem, como o Hiperescala (Citus), devem esperar erros transitórios e reagir normalmente. Por exemplo, os aplicativos devem implementar a lógica de repetição para tratar esses erros em vez de identificá-los como erros de aplicativo para os usuários.
3. Conforme um servidor se aproxima dos limites de recursos, os erros podem parecer um problema de conectividade transitório. Aumentar a RAM do nó ou adicionar nós de trabalho e reequilibrar dados pode ajudar.
4. Se os problemas de conectividade continuarem, ou durarem mais de 60 segundos, ou ocorrerem mais de uma vez por dia, preencha uma solicitação de suporte do Azure selecionando **Obter suporte** no site de [Suporte do Azure](#).

Solucionar erros persistentes

Se o aplicativo falhar de forma persistente ao se conectar com Hiperescala (Citus), as causas mais comuns são configuração incorreta do firewall ou erro do usuário.

- Configuração do firewall do nó coordenador: certifique-se que o firewall do servidor de Hiperescala (Citus) está configurado para permitir conexões do cliente, incluindo servidores proxy e gateways.
- Configuração do firewall do cliente: o firewall do cliente deve permitir conexões com o servidor de banco de dados. Alguns firewalls exigem permissão não apenas para aplicativos por nome, mas permissão para endereços IP e portas do servidor.
- Erro do usuário: verifique novamente a cadeia de conexão. Você pode ter digitado incorretamente parâmetros como o nome do servidor. Você pode encontrar cadeias de conexão para diversas estruturas de linguagem e psql no portal do Azure. Vá para a página **Cadeias de conexão** em seu grupo de servidores de Hiperescala (Citus). Além disso, tenha em mente que os clusters de Hiperescala (Citus) têm apenas um banco de dados e seu nome predefinido é **citus**.

Etapas para resolver os problemas de conectividade temporários

1. Configure as [regras de firewall](#) para permitir o endereço IP do cliente. Para fins de testes temporários, configure uma regra de firewall usando 0.0.0.0 como o endereço IP inicial e usando 255.255.255.255 como o endereço IP final. Essa regra abre o servidor para todos os endereços IP. Se a regra resolver seu problema de conectividade, remova-a e crie uma regra de firewall para um endereço IP limitado ou intervalo de endereços apropriados.
2. Em todos os firewalls entre o cliente e a Internet, verifique se a porta 5432 está aberta para conexões de saída.
3. Verifique a cadeia de conexão e outras configurações de conexão.
4. Verifique a integridade do serviço no painel.

Próximas etapas

- Aprenda os conceitos das [Regras de firewall no Banco de Dados do Azure para PostgreSQL - Hiperescala \(Citus\)](#)
- Saiba como [Gerenciar regras de firewall para o Banco de Dados do Azure para PostgreSQL - Hiperescala \(Citus\)](#)

Solucionar problemas de acesso somente leitura do Banco de Dados do Azure para PostgreSQL – Hiperescala (Citus)

09/08/2021 • 2 minutes to read

O PostgreSQL não pode executar em um computador sem espaço em disco livre. Para manter o acesso aos servidores PostgreSQL, é necessário que o espaço em disco seja suficiente.

Na Hiperescala (Citus), os nós serão configurados para um estado RO (somente leitura) quando o disco estiver quase cheio. Ao impedir as gravações, impedirá que o disco continue enchendo e manterá o nó disponível para leituras. Durante o estado somente leitura, é possível tomar medidas para liberar mais espaço em disco.

Especificamente, um nó de Hiperescala (Citus) se torna somente leitura quando tiver menos de 5 GiB de armazenamento livre restante. Quando o servidor se torna somente leitura, todas as sessões existentes são desconectadas e as transações não confirmadas são revertidas. Todas as operações de gravação e confirmações de transação falharão, enquanto as consultas de leitura continuarão funcionando.

Maneiras de recuperar o acesso de gravação

No nó coordenador

- [Aumente o tamanho do armazenamento](#) no nó coordenador e/ou
- Distribua as tabelas locais para nós de trabalho ou remova dados. Para qualquer uma das opções, será necessário executar `SET SESSION CHARACTERISTICS AS TRANSACTION READ WRITE` após conectar o banco de dados e antes de executar outros comandos.

Em um nó de trabalho

- [Aumente o tamanho do armazenamento](#) nos nós de trabalho e/ou
- [Redistribua os dados](#) para outros nós ou remova alguns dados.
 - Para qualquer uma das opções, será necessário definir temporariamente o nó de trabalho como leitura/gravação. Para fazer isso, envie uma solicitação de suporte. Como alternativa, se estiver executando um grupo de servidor de Hiperescala (Citus) de versão prévia, será possível conectar diretamente os nós de trabalho e usar `SET SESSION CHARACTERISTICS`, conforme descrito acima para o nó coordenador.

Prevenção

É recomendável configurar um alerta para notificar quando o armazenamento do servidor estiver se aproximando do limite. Dessa forma, você poderá agir antecipadamente para não entrar no estado somente leitura. Para obter mais informações, consulte a documentação sobre os [alertas recomendados](#).

Próximas etapas

- [Configurar os alertas do Azure](#) para avisar antecipadamente e possibilitar a tomada de ação antes de atingir o estado somente leitura.
- Saiba mais sobre o [uso de disco](#) na documentação do PostgreSQL.
- Saiba mais sobre as [características da sessão](#) na documentação do PostgreSQL.

Consultas de diagnóstico úteis

09/08/2021 • 5 minutes to read

Encontrando qual nó contém dados de um locatário específico

No caso de uso de vários locatários, podemos determinar qual nó de trabalho contém as linhas de um locatário específico. A Hiperescala (Citus) agrupa as linhas de tabelas distribuídas em fragmentos e coloca cada fragmento em um nó de trabalho no grupo de servidores.

Suponha que os locatários de nosso aplicativo sejam lojas e que desejamos descobrir qual nó de trabalho contém os dados da loja cuja ID = 4. Ou seja, queremos descobrir o posicionamento do fragmento que contém as linhas cuja coluna de distribuição tem o valor 4:

```
SELECT shardid, shardstate, shardlength, nodename, nodeport, placementid
  FROM pg_dist_placement AS placement,
       pg_dist_node AS node
 WHERE placement.groupid = nodegroupid
   AND node.noderole = 'primary'
   AND shardid = (
     SELECT get_shard_id_for_distribution_column('stores', 4)
   );
```

A saída contém o host e a porta do banco de dados de trabalho.

shardid	shardstate	shardlength	nodename	nodeport	placementid
102009	1	0	10.0.0.16	5432	2

Encontrando a coluna de distribuição de uma tabela

Cada tabela distribuída na Hiperescala (Citus) tem uma "coluna de distribuição". (Para saber mais, confira [Modelagem de dados distribuídos](#).) Pode ser importante saber qual é a coluna. Por exemplo, ao unir ou filtrar tabelas, você poderá ver mensagens de erro com dicas como "adicone um filtro à coluna de distribuição".

As tabelas `pg_dist_*` no nó coordenador contêm diversos metadados sobre o banco de dados distribuído. Em particular, `pg_dist_partition` contém informações sobre a coluna de distribuição para cada tabela. Você pode usar uma função de utilitário conveniente para pesquisar o nome da coluna de distribuição nos detalhes de nível inferior nos metadados. Veja um exemplo e sua saída:

```

-- create example table

CREATE TABLE products (
    store_id bigint,
    product_id bigint,
    name text,
    price money,

    CONSTRAINT products_pkey PRIMARY KEY (store_id, product_id)
);

-- pick store_id as distribution column

SELECT create_distributed_table('products', 'store_id');

-- get distribution column name for products table

SELECT column_to_column_name(logicalrelid, partkey) AS dist_col_name
  FROM pg_dist_partition
 WHERE logicalrelid='products'::regclass;

```

Saída de exemplo:

dist_col_name
store_id

Detectando bloqueios

Essa consulta será executada em todos os nós de trabalho e identificará os bloqueios, há quanto tempo eles estão abertos e as consultas incorretas:

```

SELECT run_command_on_workers($cmd$)
  SELECT array_agg(
    blocked_statement || ' ' || cur_stmt_blocking_proc
    || ' ' || cnt::text || ' ' || age
  )
  FROM (
    SELECT blocked_activity.query AS blocked_statement,
           blocking_activity.query AS cur_stmt_blocking_proc,
           count(*) AS cnt,
           age(now(), min(blocked_activity.query_start)) AS "age"
      FROM pg_catalog.pg_locks blocked_locks
     JOIN pg_catalog.pg_stat_activity blocked_activity
       ON blocked_activity.pid = blocked_locks.pid
     JOIN pg_catalog.pg_locks blocking_locks
       ON blocking_locks.locktype = blocked_locks.locktype
      AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
      AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
      AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
      AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
      AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
      AND blocking_locks.transactionid IS NOT DISTINCT FROM blocked_locks.transactionid
      AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
      AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
      AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
      AND blocking_locks.pid != blocked_locks.pid
     JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid = blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED
      AND blocking_locks.GRANTED
    GROUP BY blocked_activity.query,
             blocking_activity.query
   ORDER BY 4
  ) a
$cmd$;

```

Saída de exemplo:

run_command_on_workers
(10.0.0.16,5432,t,"")
(10.0.0.20,5432,t,"{"update ads_102277 set name = 'new name' where id = 1; \$ select * from ads_102277 where id = 1 for update; \$ 1 \$ 00:00:03.729519"}")

Consultar o tamanho de seus fragmentos

Essa consulta fornecerá o tamanho de cada fragmento de uma determinada tabela distribuída, chamada

`my_distributed_table`:

```

SELECT *
  FROM run_command_on_shards('my_distributed_table', $cmd$
  SELECT json_build_object(
    'shard_name', '%1$s',
    'size',      pg_size_pretty(pg_table_size('%1$s'))
  );
$cmd$);

```

Saída de exemplo:

shardid	success	result
102008	t	{"shard_name" : "my_distributed_table_102008", "size" : "2416 kB"}
102009	t	{"shard_name" : "my_distributed_table_102009", "size" : "3960 kB"}
102010	t	{"shard_name" : "my_distributed_table_102010", "size" : "1624 kB"}
102011	t	{"shard_name" : "my_distributed_table_102011", "size" : "4792 kB"}

Consultando o tamanho de todas as tabelas distribuídas

Essa consulta obtém uma lista dos tamanhos de cada tabela distribuída e o tamanho de seus índices.

```
SELECT
    tablename,
    pg_size_pretty(
        citus_total_relation_size(tablename::text)
    ) AS total_size
FROM pg_tables pt
JOIN pg_dist_partition pp
    ON pt.tablename = pp.logicalrelid::text
WHERE schemaname = 'public';
```

Saída de exemplo:

tablename	total_size
github_users	39 MB
github_events	98 MB

Observe que há outras funções da Hiperescala (Citus) para consultar o tamanho da tabela distribuída; confira [determinando o tamanho da tabela](#).

Identificar índices não utilizados

A consulta a seguir identificará índices não utilizados em nós de trabalho para uma determinada tabela distribuída (`my_distributed_table`)

```

SELECT *
FROM run_command_on_shards('my_distributed_table', $cmd$)
  SELECT array_agg(a) as infos
  FROM (
    SELECT (
      schemaname || '.' || relname || '##' || indexrelname || '##'
        || pg_size.pretty(pg_relation_size(i.indexrelid))::text
        || '##' || idx_scan::text
    ) AS a
  FROM pg_stat_user_indexes ui
  JOIN pg_index i
  ON ui.indexrelid = i.indexrelid
  WHERE NOT indisunique
  AND idx_scan < 50
  AND pg_relation_size(relid) > 5 * 8192
  AND (schemaname || '.' || relname)::regclass = '%s'::regclass
  ORDER BY
    pg_relation_size(i.indexrelid) / NULLIF(idx_scan, 0) DESC nulls first,
    pg_relation_size(i.indexrelid) DESC
  ) sub
$cmd$;

```

Saída de exemplo:

shardid	success	result
102008	t	
102009	t	{"public.my_distributed_table_102009##some_index_102009##28 MB##0"}
102010	t	
102011	t	

Monitorando a contagem de conexões do cliente

A consulta a seguir conta as conexões abertas no coordenador e as agrupa por tipo.

```

SELECT state, count(*)
FROM pg_stat_activity
GROUP BY state;

```

Saída de exemplo:

state	count
active	3
idle	3
∅	6

Taxa de ocorrência no índice

Essa consulta fornecerá sua taxa de ocorrência no índice em todos os nós. A taxa de ocorrência no índice é útil para determinar com que frequência os índices são usados ao consultar:

```

SELECT nodename, result as index_hit_rate
FROM run_command_on_workers($cmd$
  SELECT CASE sum(idx_blk_hit)
    WHEN 0 THEN 'NaN'::numeric
    ELSE to_char((sum(idx_blk_hit) - sum(idx_blk_read)) / sum(idx_blk_hit + idx_blk_read),
    '99.99')::numeric
    END AS ratio
  FROM pg_statio_user_indexes
$cmd$);

```

Saída de exemplo:

nodename	index_hit_rate
10.0.0.16	0.88
10.0.0.20	0.89

Taxa de ocorrência no cache

A maioria dos aplicativos costuma acessar uma pequena fração de seu total de dados por vez. O PostgreSQL mantém dados acessados com frequência na memória para evitar leituras de disco lentas. Veja estatísticas sobre ele na exibição [pg_statio_user_tables](#).

Uma medida importante é qual percentual dos dados é proveniente do cache na memória versus o disco em sua carga de trabalho:

```

SELECT
  sum(heap_blk_read) AS heap_read,
  sum(heap_blk_hit) AS heap_hit,
  sum(heap_blk_hit) / (sum(heap_blk_hit) + sum(heap_blk_read)) AS ratio
FROM
  pg_statio_user_tables;

```

Saída de exemplo:

heap_read	heap_hit	ratio
1	132	0.99248120300751879699

Se você tem uma taxa significativamente inferior a 99%, considere aumentar o cache disponível para seu banco de dados.

Próximas etapas

- Saiba mais sobre outras [tabelas do sistema](#) que são úteis para diagnóstico

Funções na API do SQL de Hiperescala (Citus)

09/08/2021 • 21 minutes to read

Esta seção contém informações de referência sobre as funções definidas pelo usuário fornecidas pela Hiperescala (Citus). Essas funções ajudam no fornecimento de funcionalidade distribuída para a Hiperescala (Citus).

NOTE

Os grupos de servidores de Hiperescala (Citus) que executam versões mais antigas do mecanismo Citus podem não oferecer todas as funções listadas abaixo.

DDL de fragmento e tabela

`create_distributed_table`

A função `create_distributed_table()` é usada para definir uma tabela distribuída e criar os fragmentos, caso ela seja uma tabela distribuída por hash. Essa função usa um nome de tabela, a coluna de distribuição e um método de distribuição opcional e insere os metadados apropriados para marcar a tabela como distribuída. A função usa como padrão a distribuição 'hash' se nenhum método de distribuição for especificado. Se a tabela for distribuída por hash, a função também criará fragmentos de trabalho com base nos valores de configuração de fator de replicação de fragmento e contagem de fragmentos. Se a tabela contiver linhas, elas serão distribuídas automaticamente aos nós de trabalho.

Essa função substitui o uso de `master_create_distributed_table()` seguido por `master_create_worker_shards()`.

Argumentos

table_name: o nome da tabela que precisa ser distribuída.

distribution_column: a coluna na qual a tabela deve ser distribuída.

distribution_type: (opcional) o método de acordo com o qual a tabela deve ser distribuída. Os valores permitidos são append ou hash, com um valor padrão de 'hash'.

colocate_with: (opcional) inclui a tabela atual no grupo de colocação de outra tabela. Por padrão, as tabelas são colocadas quando são distribuídas por colunas do mesmo tipo, têm a mesma contagem de fragmentos e o mesmo fator de replicação. Os valores possíveis para `colocate_with` são `default`, `none`, que inicia um novo grupo de colocação, ou o nome de outra tabela a ser colocada com essa tabela. (Confira [colocação de tabela](#).)

Tenha em mente que o valor padrão de `colocate_with` realiza a colocação implícita. A colocação pode ser positiva quando as tabelas estão relacionadas ou serão unidas. No entanto, quando duas tabelas não são relacionadas, mas usam o mesmo tipo de dados nas colunas de distribuição, a colocação acidental delas pode diminuir o desempenho durante o [rebalanceamento de fragmento](#). Os fragmentos de tabela serão movidos juntos desnecessariamente em uma "cascata."

Se uma nova tabela distribuída não é relacionada a outras tabelas, é melhor especificar

```
colocate_with => 'none'.
```

Valor Retornado

N/D

Exemplo

Este exemplo informa ao banco de dados que a tabela `github_events` deve ser distribuída por hash na coluna

repo_id.

```
SELECT create_distributed_table('github_events', 'repo_id');

-- alternatively, to be more explicit:
SELECT create_distributed_table('github_events', 'repo_id',
                               colocate_with => 'github_repo');
```

create_reference_table

A função `create_reference_table()` é usada para definir uma pequena referência ou tabela de dimensões. Essa função usa um nome de tabela e cria uma tabela distribuída com apenas um fragmento, replicado para cada nó de trabalho.

Argumentos

`table_name`: o nome da dimensão pequena ou da tabela de referência que precisa ser distribuída.

Valor Retornado

N/D

Exemplo

Este exemplo informa ao banco de dados que a tabela nation deve ser definida como uma tabela de referência

```
SELECT create_reference_table('nation');
```

upgrade_to_reference_table

A função `upgrade_to_reference_table()` usa uma tabela distribuída existente que tem uma contagem de fragmentos equivalente a um e a atualiza para torná-la uma tabela de referência reconhecida. Depois de chamar essa função, será como se a tabela tivesse sido criada com [create_reference_table](#).

Argumentos

`table_name`: o nome da tabela distribuída (com a contagem de fragmentos = 1) que será distribuída como uma tabela de referência.

Valor Retornado

N/D

Exemplo

Este exemplo informa ao banco de dados que a tabela nation deve ser definida como uma tabela de referência

```
SELECT upgrade_to_reference_table('nation');
```

mark_tables_colocated

A função `mark_tables_colocated()` usa uma tabela distribuída (a origem) e uma lista de outras tabelas (os destinos) e coloca os destinos no mesmo grupo de colocação que a origem. Se a origem ainda não estiver em um grupo, essa função criará um e atribuirá a origem e os destinos a ele.

A colocação de tabelas deve ser feita no tempo de distribuição da tabela por meio do parâmetro `colocate_with` de [create_distributed_table](#), mas `mark_tables_colocated` pode realizar isso posteriormente, se necessário.

Argumentos

`source_table_name`: o nome da tabela distribuída cujo grupo de colocações será atribuído aos destinos para correspondência.

`target_table_names`: a matriz de nomes das tabelas de destino distribuídas não pode ser vazia. Essas tabelas distribuídas precisam corresponder à tabela de origem em:

- Método distribution
- tipo de coluna de distribuição
- tipo de replicação
- contagem de fragmento

Se nenhuma das alternativas acima é aplicável, a Hiperescala (Citus) gera um erro. Por exemplo, a tentativa de colocar as tabelas `apples` e `oranges` cujos tipos de coluna de distribuição são diferentes resulta em:

```
ERROR: XX000: cannot colocate tables apples and oranges
DETAIL: Distribution column types don't match for apples and oranges.
```

Valor Retornado

N/D

Exemplo

Este exemplo coloca `products` e `line_items` no mesmo grupo de colocação que `stores`. O exemplo supõe que essas tabelas sejam todas distribuídas em uma coluna com tipo correspondente, provavelmente uma "ID de loja."

```
SELECT mark_tables_colocated('stores', ARRAY['products', 'line_items']);
```

`create_distributed_function`

Propaga uma função do nó coordenador para os trabalhos e marca-a para execução distribuída. Quando uma função distribuída é chamada no coordenador, a Hiperescala (Citus) usa o valor do "argumento de distribuição" para escolher um nó de trabalho para executar a função. A execução da função nos trabalhos aumenta o paralelismo e pode trazer o código mais próximo aos dados em fragmentos para uma latência mais baixa.

O caminho de pesquisa do Postgres não é propagado do coordenador para os trabalhos durante a execução da função distribuída, portanto, o código de função distribuído deve qualificar totalmente os nomes dos objetos de banco de dados. Além disso, as notificações emitidas pelas funções não serão exibidos para o usuário.

Argumentos

function_name: o nome da função a ser distribuída. O nome precisa incluir os tipos de parâmetro da função entre parênteses, pois várias funções podem ter o mesmo nome no PostgreSQL. Por exemplo, `'foo(int)'` é diferente de `'foo(int, text)'`.

distribution_arg_name: (opcional) o nome do argumento pelo qual distribuir. Para maior conveniência (ou se os argumentos de função não tiverem nomes), um espaço reservado posicional é permitido, como `'$1'`. Se esse parâmetro não for especificado, a função nomeada por `function_name` será simplesmente criada nos trabalhos. Se os nós de trabalho forem adicionados no futuro, a função será automaticamente criada neles.

colocate_with: (opcional) quando a função distribuída lê ou grava em uma tabela distribuída (ou, mais comumente, no grupo de colocação), nomeie essa tabela usando o parâmetro `colocate_with`. Em seguida, cada invocação da função será executada no nó de trabalho que contém os fragmentos relevantes.

Valor Retornado

N/D

Exemplo

```

-- an example function which updates a hypothetical
-- event_responses table which itself is distributed by event_id
CREATE OR REPLACE FUNCTION
    register_for_event(p_event_id int, p_user_id int)
RETURNS void LANGUAGE plpgsql AS $fn$
BEGIN
    INSERT INTO event_responses VALUES ($1, $2, 'yes')
    ON CONFLICT (event_id, user_id)
    DO UPDATE SET response = EXCLUDED.response;
END;
$fn$;

-- distribute the function to workers, using the p_event_id argument
-- to determine which shard each invocation affects, and explicitly
-- colocating with event_responses which the function updates
SELECT create_distributed_function(
    'register_for_event(int, int)', 'p_event_id',
    colocate_with := 'event_responses'
);

```

alter_columnar_table_set

A função `alter_columnar_table_set()` altera as configurações na [tabela em coluna](#). Chamar essa função em uma tabela sem coluna gera um erro. Todos os argumentos, exceto o nome da tabela, são opcionais.

Para exibir as opções atuais de todas as tabelas em coluna, confira esta tabela:

```
SELECT * FROM columnar.options;
```

Os valores padrão das configurações de coluna para tabelas recém-criadas podem ser substituídos por estes GUCs:

- `columnar.compression`
- `columnar.compression_level`
- `columnar.stripe_row_count`
- `columnar.chunk_row_count`

Argumentos

table_name: nome da tabela em coluna.

chunk_row_count: (opcional) o número máximo de linhas por parte para dados recém-inseridos. As partes de dados existentes não serão alteradas e poderão ter mais linhas do que esse valor máximo. O valor padrão é 10000.

stripe_row_count: (opcional) o número máximo de linhas por faixas para dados recém-inseridos. As faixas de dados existentes não serão alteradas e poderão ter mais linhas do que esse valor máximo. O valor padrão é 150000.

compression: (opcional) `[none|pglz|zstd|lz4|lz4hc]` o tipo de compactação para dados recém-inseridos. Os dados existentes não serão compactados nem descompactados. O valor padrão e sugerido é zstd (se o suporte tiver sido compilado).

compression_level: (opcional) as configurações válidas são de 1 a 19. Se o método de compactação não dá suporte ao nível escolhido, o nível mais próximo será selecionado.

Retornar valor

N/D

Exemplo

```
SELECT alter_columnar_table_set(
    'my_columnar_table',
    compression => 'none',
    stripe_row_count => 10000);
```

Informações de metadados/configuração

master_get_table_metadata

A função `master_get_table_metadata()` pode ser usada para retornar metadados relacionados à distribuição de uma tabela distribuída. Esses metadados incluem a ID da relação, o tipo de armazenamento, o método de distribuição, a coluna de distribuição, a contagem de replicação, o tamanho máximo do fragmento e a política de posicionamento do fragmento para a tabela. Nos bastidores, essa função consulta as tabelas de metadados da Hiperescala (Citus) para obter as informações necessárias e concatena-as em uma tupla antes de retorná-las ao usuário.

Argumentos

`table_name`: o nome da tabela distribuída para a qual você deseja buscar metadados.

Valor Retornado

Uma tupla que contém as seguintes informações:

`logical_relid`: o OID da tabela distribuída. Ele faz referência à coluna `reloid` na tabela de catálogo do sistema `pg_class`.

`part_storage_type`: tipo de armazenamento usado para a tabela. Pode ser '`t`' (tabela padrão), '`f`' (tabela estrangeira) ou '`c`' (tabela de coluna).

`part_method`: método de distribuição usado na tabela. Pode ser '`a`' (acrescentar) ou '`h`' (hash).

`part_key`: coluna de distribuição da tabela.

`part_replica_count`: contagem de replicação de fragmento atual.

`part_max_size`: tamanho máximo atual do fragmento em bytes.

`part_placement_policy`: política de posicionamento de fragmento usada para colocar os fragmentos da tabela. Pode ser 1 (local-node-first) ou 2 (round-robin).

Exemplo

O exemplo a seguir busca e exibe os metadados da tabela `github_events`.

```
SELECT * from master_get_table_metadata('github_events');
 logical_relid | part_storage_type | part_method | part_key | part_replica_count | part_max_size |
 part_placement_policy
-----+-----+-----+-----+-----+-----+
 24180 | t           | h           | repo_id | 2             | 1073741824 |
-----+-----+-----+-----+-----+-----+
 2
(1 row)
```

get_shard_id_for_distribution_column

A Hiperescala (Citus) atribui todas as linhas de uma tabela distribuída a um fragmento com base no valor da coluna de distribuição da linha e no método de distribuição da tabela. Na maioria dos casos, o mapeamento preciso é um detalhe de baixo nível que o administrador de banco de dados pode ignorar. No entanto, pode ser útil determinar o fragmento de uma linha para tarefas manuais de manutenção de banco de dados ou apenas por curiosidade. A função `get_shard_id_for_distribution_column` fornece essas informações de tabelas distribuídas por hash e por intervalo e tabelas de referência. Ela não funciona na distribuição de acréscimo.

Argumentos

table_name: a tabela distribuída.

distribution_value: o valor da coluna de distribuição.

Valor Retornado

A Hiperescala (Citus) de ID de fragmento se associa ao valor da coluna de distribuição da tabela especificada.

Exemplo

```
SELECT get_shard_id_for_distribution_column('my_table', 4);

get_shard_id_for_distribution_column
-----
      540007
(1 row)
```

column_to_column_name

Move a coluna `partkey` de `pg_dist_partition` para um nome de coluna textual. Essa movimentação é útil para determinar a coluna de distribuição de uma tabela distribuída.

Para ver uma discussão mais aprofundada, confira [como escolher uma coluna de distribuição](#).

Argumentos

table_name: a tabela distribuída.

column_var_text: o valor de `partkey` na tabela `pg_dist_partition`.

Valor Retornado

O nome da coluna de distribuição de `table_name`.

Exemplo

```
-- get distribution column name for products table

SELECT column_to_column_name(logicalrelid, partkey) AS dist_col_name
  FROM pg_dist_partition
 WHERE logicalrelid='products'::regclass;
```

Saída:

dist_col_name
company_id

citus_relation_size

Obtenha o espaço em disco usado por todos os fragmentos da tabela distribuída especificada. O espaço em disco inclui o tamanho da "bifurcação principal," mas exclui o mapa de visibilidade e o mapa de espaço livre dos fragmentos.

Argumentos

logicalrelid: o nome de uma tabela distribuída.

Valor Retornado

Tamanho em bytes como um bigint.

Exemplo

```
SELECT pg_size.pretty(citus_relation_size('github_events'));
```

```
pg_size.pretty
-----
23 MB
```

citus_table_size

Obtenha o espaço em disco usado por todos os fragmentos da tabela distribuída especificada, excluindo os índices (mas incluindo a notificação do sistema, o mapa de espaço livre e o mapa de visibilidade).

Argumentos

logicalrelid: o nome de uma tabela distribuída.

Valor Retornado

Tamanho em bytes como um bigint.

Exemplo

```
SELECT pg_size.pretty(citus_table_size('github_events'));
```

```
pg_size.pretty
-----
37 MB
```

citus_total_relation_size

Obtenha o espaço em disco total usado por todos os fragmentos da tabela distribuída especificada, incluindo todos os índices e dados de notificação do sistema.

Argumentos

logicalrelid: o nome de uma tabela distribuída.

Valor Retornado

Tamanho em bytes como um bigint.

Exemplo

```
SELECT pg_size.pretty(citus_total_relation_size('github_events'));
```

```
pg_size.pretty
-----
73 MB
```

citus_stat_statements_reset

Remove todas as linhas de [citus_stat_statements](#). Essa função funciona independentemente do `pg_stat_statements_reset()`. Para redefinir todas as estatísticas, chame ambas as funções.

Argumentos

N/D

Valor Retornado

Nenhum

Gerenciamento e reparo do grupo de servidores

master_copy_shard_placement

Se um posicionamento de fragmento não for atualizado durante um comando de modificação ou em uma operação de DDL, ele será marcado como inativo. A função master_copy_shard_placement pode ser chamada para reparar um posicionamento de fragmento inativo usando dados de um posicionamento íntegro.

Para reparar um fragmento, primeiro a função remove o posicionamento do fragmento não íntegro e o recria usando o esquema no coordenador. Depois que o posicionamento do fragmento é criado, a função copia dados do posicionamento íntegro e atualiza os metadados para marcar o novo posicionamento do fragmento como íntegro. Essa função garante que o fragmento será protegido contra qualquer modificação simultânea durante o reparo.

Argumentos

shard_id: a ID do fragmento que será reparado.

source_node_name: o nome DNS do nó no qual o posicionamento do fragmento íntegro está presente (nó de "origem").

source_node_port: a porta no nó de trabalho de origem no qual o servidor de banco de dados está escutando.

target_node_name: o nome DNS do nó no qual o posicionamento de fragmento inválido está presente (nó de "destino").

target_node_port: a porta no nó de trabalho de destino no qual o servidor de banco de dados está escutando.

Valor Retornado

N/D

Exemplo

O exemplo a seguir repara um posicionamento de fragmento inativo do fragmento 12345, que está presente no servidor de banco de dados que está sendo executado em 'bad_host' na porta 5432. Para repará-lo, ele usará dados de um posicionamento de fragmento íntegro presente no servidor que está sendo executado no 'good_host' na porta 5432.

```
SELECT master_copy_shard_placement(12345, 'good_host', 5432, 'bad_host', 5432);
```

master_move_shard_placement

Essa função move um determinado fragmento (e os fragmentos colocados com ele) de um nó para outro. Normalmente, ele é usado indiretamente durante o rebalanceamento de fragmento em vez de ser chamado diretamente por um administrador de banco de dados.

Há duas maneiras de mover os dados: com bloqueio ou sem bloqueio. A abordagem com bloqueio significa que, durante a movimentação, todas as modificações no fragmento são colocadas em pausa. A segunda maneira, que evita o bloqueio de gravações de fragmento, depende da replicação lógica do Postgres 10.

Após uma operação de movimentação bem-sucedida, os fragmentos no nó de origem são excluídos. Se a movimentação tem uma falha a qualquer momento, essa função gera um erro e deixa os nós de origem e de destino inalterados.

Argumentos

shard_id: a ID do fragmento que será movido.

source_node_name: o nome DNS do nó no qual o posicionamento do fragmento íntegro está presente (nó de "origem").

source_node_port: a porta no nó de trabalho de origem no qual o servidor de banco de dados está escutando.

target_node_name: o nome DNS do nó no qual o posicionamento de fragmento inválido está presente (nó de "destino").

target_node_port: a porta no nó de trabalho de destino no qual o servidor de banco de dados está escutando.

shard_transfer_mode: (opcional) especifique o método de replicação, se a replicação lógica do PostgreSQL deve ser usada ou um comando COPY entre trabalhos. Os valores possíveis são:

- `auto` : exigirá a identidade de réplica se a replicação lógica for possível; caso contrário, usará o comportamento herdado (por exemplo, para reparo de fragmentos no PostgreSQL 9.6). Esse é o valor padrão.
- `force_logical` : use a replicação lógica, mesmo que a tabela não tenha uma identidade de réplica. Todas as instruções de atualização/exclusão simultâneas para a tabela falharão durante a replicação.
- `block_writes` : use COPY (bloqueio de gravações) em tabelas que não têm a chave primária ou a identidade de réplica.

Valor Retornado

N/D

Exemplo

```
SELECT master_move_shard_placement(12345, 'from_host', 5432, 'to_host', 5432);
```

rebalance_table_shards

A função `rebalance_table_shards()` move fragmentos da tabela fornecida para torná-los uniformemente distribuídos entre os trabalhos. Primeiro a função calcula a lista de movimentações que precisa fazer para garantir que o grupo de servidores seja balanceado dentro do limite fornecido. Em seguida, ele move os posicionamentos de fragmentos um de cada vez do nó de origem para o nó de destino e atualiza os metadados de fragmento correspondentes para refletir essa movimentação.

Cada fragmento recebe um custo ao determinar se os fragmentos são "distribuídos uniformemente." Por padrão, cada fragmento tem o mesmo custo (um valor de 1), portanto, a distribuição para equalizar o custo nos trabalhos é igual à equalização do número de fragmentos em cada trabalho. A estratégia de custo constante é chamada de "by_shard_count" e é a estratégia de rebalanceamento padrão.

A estratégia padrão é apropriada nas seguintes circunstâncias:

- Os fragmentos são aproximadamente do mesmo tamanho
- Os fragmentos recebem aproximadamente a mesma quantidade de tráfego
- Os nós de trabalho são todos do mesmo tamanho/tipo
- Os fragmentos não foram fixados a determinados trabalhos

Se qualquer uma dessas suposições não for verdadeira, o rebalanceamento padrão poderá resultar em um plano inadequado. Nesse caso, você pode personalizar a estratégia usando o parâmetro `rebalance_strategy`.

É aconselhável chamar `get_rebalance_table_shards_plan` antes de executar `rebalance_table_shards`, para ver e verificar as ações a serem executadas.

Argumentos

table_name: (opcional) o nome da tabela cujos fragmentos precisam ser rebalanceados. Se for NULL, balanceie novamente todos os grupos de colocação existentes.

threshold: (Opcional) um número flutuante entre 0,0 e 1,0 que indica a razão de diferença máxima de utilização de nó por meio da utilização média. Por exemplo, especificar 0,1 fará com que o rebalanceador de fragmentos tente balancear todos os nós para manter o mesmo número de fragmentos a ±10%.

Especificamente, o rebalanceador de fragmentos tentará convergir a utilização de todos os nós de trabalho para a utilização média $(1 - \text{limite}) * \text{utilização_média} \dots (1 - \text{limite}) * \text{intervalo médio de utilização}$.

- `limite` * intervalo médio de utilização.

max_shard_moves: (opcional) o número máximo de fragmentos a serem movidos.

excluded_shard_list: (opcional) identificadores de fragmentos que não devem ser movidos durante a operação de rebalanceamento.

shard_transfer_mode: (opcional) especifique o método de replicação, se a replicação lógica do PostgreSQL deve ser usada ou um comando COPY entre trabalhos. Os valores possíveis são:

- `auto` : exigirá a identidade de réplica se a replicação lógica for possível; caso contrário, usará o comportamento herdado (por exemplo, para reparo de fragmentos no PostgreSQL 9.6). Esse é o valor padrão.
- `force_logical` : use a replicação lógica, mesmo que a tabela não tenha uma identidade de réplica. Todas as instruções de atualização/exclusão simultâneas para a tabela falharão durante a replicação.
- `block_writes` : use COPY (bloqueio de gravações) em tabelas que não têm a chave primária ou a identidade de réplica.

drain_only: (opcional) quando é definido como true, ele retira os fragmentos de nós de trabalho que têm `shouldhaveshards` definido como false no [pg_dist_node](#). Não move nenhum outro fragmento.

rebalance_strategy: (opcional) o nome de uma estratégia em [pg_dist_rebalance_strategy](#). Se esse argumento for omitido, a função escolherá a estratégia padrão, conforme indicado na tabela.

Valor Retornado

N/D

Exemplo

O exemplo a seguir tentará rebalancear os fragmentos da tabela `github_events` dentro do limite padrão.

```
SELECT rebalance_table_shards('github_events');
```

Este exemplo de uso tentará rebalancear a tabela `github_events` sem mover fragmentos com a ID 1 e 2.

```
SELECT rebalance_table_shards('github_events', excluded_shard_list:='{}1,2{}');
```

get_rebalance_table_shards_plan

Gere os movimentos do fragmento planejados de [rebalance_table_shards](#) sem executá-los. Embora seja improvável, o `get_rebalance_table_shards_plan` pode gerar um plano um pouco diferente do que uma chamada `rebalance_table_shards` com os mesmos argumentos. Eles não são executados ao mesmo tempo, portanto, os fatos sobre o grupo de servidores, -por exemplo, espaço em disco,- podem ser diferentes entre as chamadas.

Argumentos

Os mesmos argumentos que `rebalance_table_shards`: `relation`, `threshold`, `max_shard_moves`, `excluded_shard_list` e `drain_only`. Confira a documentação dessa função para obter o significado dos argumentos.

Valor Retornado

Tuplas que contêm estas colunas:

- `table_name`: a tabela cujos fragmentos seriam movidos
- `shardid`: o fragmento em questão
- `shard_size`: o tamanho em bytes

- **sourcename**: o nome do host do nó de origem
- **sourceport**: a porta do nó de origem
- **targetname**: o nome do host do nó de destino
- **targetport**: a porta do nó de destino

get_rebalance_progress

Quando um rebalanceamento de fragmentos começa, a função `get_rebalance_progress()` lista o progresso de cada fragmento envolvido. Ele monitora as movimentações planejadas e executadas pelo `rebalance_table_shards()`.

Argumentos

N/D

Valor Retornado

Tuplas que contêm estas colunas:

- **sessionid**: o PID do Postgres do monitor de rebalanceamento
- **table_name**: a tabela cujos fragmentos estão sendo movidos
- **shardid**: o fragmento em questão
- **shard_size**: o tamanho em bytes
- **sourcename**: o nome do host do nó de origem
- **sourceport**: a porta do nó de origem
- **targetname**: o nome do host do nó de destino
- **targetport**: a porta do nó de destino
- **progress**: 0 = aguardando ser movido; 1 = sendo movido; 2 = concluído

Exemplo

```
SELECT * FROM get_rebalance_progress();
```

sessionid	table_name	shardid	shard_size	sourcename	sourceport	targetname	targetport
7083	foo	102008	1204224	n1.foobar.com	5432	n4.foobar.com	5432
0							
7083	foo	102009	1802240	n1.foobar.com	5432	n4.foobar.com	5432
0							
7083	foo	102018	614400	n2.foobar.com	5432	n4.foobar.com	5432
1							
7083	foo	102019	8192	n3.foobar.com	5432	n4.foobar.com	5432
2							

citus_add_rebalance_strategy

Acrescentar uma linha a `pg_dist_rebalance_strategy`.

Argumentos

Para obter mais informações sobre esses argumentos, confira os valores de coluna correspondentes em `pg_dist_rebalance_strategy`.

name: o identificador para a nova estratégia

shard_cost_function: identifica a função usada para determinar o "custo" de cada fragmento

node_capacity_function: identifica a função usada para medir a capacidade do nó

shard_allowed_on_node_function: identifica a função que determina quais fragmentos podem ser colocados em quais nós

default_threshold: um limite de ponto flutuante que ajusta o quanto precisamente o custo de fragmento cumulativo deve ser balanceado entre os nós

minimum_threshold: (opcional) uma coluna de garantia que contém o valor mínimo permitido para o argumento de limite de rebalance_table_shards(). O valor padrão é 0

Valor Retornado

N/D

citus_set_default_rebalance_strategy

Atualize a tabela [pg_dist_rebalance_strategy](#), alterando a estratégia nomeada pelo argumento como o padrão escolhido ao rebalancear fragmentos.

Argumentos

nome: o nome da estratégia em pg_dist_rebalance_strategy

Valor Retornado

N/D

Exemplo

```
SELECT citus_set_default_rebalance_strategy('by_disk_size');
```

citus_remote_connection_stats

A função citus_remote_connection_stats() mostra o número de conexões ativas para cada nó remoto.

Argumentos

N/D

Exemplo

```
SELECT * from citus_remote_connection_stats();
```

hostname	port	database_name	connection_count_to_node
citus_worker_1	5432	postgres	3

(1 row)

master_drain_node

A função master_drain_node() retira fragmentos do nó designado e os coloca em outros nós que tenham `shouldhaveshards` definido como true no [pg_dist_node](#). Chame a função antes de remover um nó do grupo de servidores e desligar o servidor físico do nó.

Argumentos

nodename: o nome do host do nó a ser esvaziado.

nodeport: o número da porta do nó a ser esvaziada.

shard_transfer_mode: (opcional) especifique o método de replicação, se a replicação lógica do PostgreSQL deve ser usada ou um comando COPY entre trabalhos. Os valores possíveis são:

- `auto` : exigirá a identidade de réplica se a replicação lógica for possível; caso contrário, usará o comportamento herdado (por exemplo, para reparo de fragmentos no PostgreSQL 9.6). Esse é o valor

padrão.

- `force_logical`: use a replicação lógica, mesmo que a tabela não tenha uma identidade de réplica. Todas as instruções de atualização/exclusão simultâneas para a tabela falharão durante a replicação.
- `block_writes`: use COPY (bloqueio de gravações) em tabelas que não têm a chave primária ou a identidade de réplica.

rebalance_strategy: (opcional) o nome de uma estratégia em [pg_dist_rebalance_strategy](#). Se esse argumento for omitido, a função escolherá a estratégia padrão, conforme indicado na tabela.

Valor Retornado

N/D

Exemplo

Veja abaixo as etapas típicas para remover um nó (por exemplo, '10.0.0.1' em uma porta PostgreSQL padrão):

1. Esvazie o nó.

```
SELECT * from master_drain_node('10.0.0.1', 5432);
```

2. Aguarde até que o comando seja finalizado

3. Remova o nó

Ao esvaziar vários nós, é recomendável usar [rebalance_table_shards](#). Isso permite que a Hiperescala (Citus) planeje antecipadamente e mova fragmentos um número mínimo de vezes.

1. Execute-o em cada nó que você deseja remover:

```
SELECT * FROM master_set_node_property(node_hostname, node_port, 'shouldhaveshards', false);
```

2. Esvazie todos de uma vez com [rebalance_table_shards](#)

```
SELECT * FROM rebalance_table_shards(drain_only := true);
```

3. Aguarde até que o rebalanceamento de esvaziamento seja concluído

4. Remova os nós

replicate_table_shards

A função `replicate_table_shards()` replica os fragmentos com baixa replicação da tabela especificada. Primeiro a função calcula a lista de fragmentos e localizações com baixa replicação nos quais eles podem ser buscados para replicação. Em seguida, a função copia esses fragmentos e atualiza os metadados de fragmento correspondentes para refletir a cópia.

Argumentos

table_name: o nome da tabela cujos fragmentos precisam ser replicados.

shard_replication_factor: (opcional) o fator de replicação desejado que deve ser alcançado em cada fragmento.

max_shard_copies: (opcional) número máximo de fragmentos a serem copiados para alcançar o fator de replicação desejado.

excluded_shard_list: (opcional) identificadores de fragmentos que não devem ser copiados durante a operação de replicação.

Valor Retornado

N/D

Exemplos

O exemplo a seguir tentará replicar os fragmentos da tabela `github_events` para o `shard_replication_factor`.

```
SELECT replicate_table_shards('github_events');
```

Este exemplo tentará trazer os fragmentos da tabela de `github_events` para o fator de replicação desejado com, no máximo, dez cópias de fragmento. O rebalanceador copiará, no máximo, dez fragmentos na tentativa de alcançar o fator de replicação desejado.

```
SELECT replicate_table_shards('github_events', max_shard_copies:=10);
```

isolate_tenant_to_new_shard

Essa função cria um fragmento para armazenar linhas com um valor específico na coluna de distribuição. Ela é especialmente útil para o caso de uso de Hiperescala (Citus) de multilocatário, em que um locatário grande pode ser colocado sozinho no próprio fragmento e, em última instância, no próprio nó físico.

Argumentos

table_name: o nome da tabela na qual será obtido um novo fragmento.

tenant_id: o valor da coluna de distribuição que será atribuída ao novo fragmento.

cascade_option: (opcional) quando definido como "CASCADE," também isola um fragmento de todas as tabelas no [grupo de colocação](#) da tabela atual.

Valor Retornado

shard_id: a função retorna a ID exclusiva atribuída ao fragmento recém-criado.

Exemplos

Crie um fragmento para manter o `lineitems` do locatário 135:

```
SELECT isolate_tenant_to_new_shard('lineitem', 135);
```

isolate_tenant_to_new_shard
102240

Próximas etapas

- Muitas das funções neste artigo modificam as [tabelas de metadados](#) do sistema
- Os [parâmetros de servidor](#) personalizam o comportamento de algumas funções

Parâmetros do Servidor

21/05/2021 • 23 minutes to read

Há vários parâmetros de servidor que afetam o comportamento da Hiperescala (Citus), do PostgreSQL padrão e da Hiperescala (Citus). Esses parâmetros podem ser definidos no portal do Azure para um grupo de servidores de Hiperescala (Citus). Na categoria **Configurações**, escolha **Parâmetros de nó de trabalho** ou **Parâmetros de nó coordenador**. Essas páginas permitem definir parâmetros para todos os nós de trabalho ou apenas para o nó coordenador.

Parâmetros de Hiperescala (Citus)

NOTE

Os grupos de servidores de Hiperescala (Citus) que executam versões mais antigas do mecanismo Citus podem não oferecer todos os parâmetros listados abaixo.

Configuração geral

citus.use_secondary_nodes (enumeração)

Define a política a ser usada ao escolher nós para consultas SELECT. Se estiver definida como 'always', o planejador consultará somente os nós marcados como 'secondary' noderole em [pg_dist_node](#).

Os valores suportados para esta enumeração são:

- **never**: (padrão) todas as leituras acontecem em nós primários.
- **always**: em vez disso, as leituras são executadas em nós secundários e as instruções insert/update são desabilitadas.

citus.cluster_name (texto)

Informa ao planejador do nó coordenador qual grupo ele coordena. Depois que _ o nome do cluster for definido, o planejador consultará os nós de trabalho apenas nesse cluster.

citus.enable_version_checks (booliano)

Atualizar a versão da Hiperescala (Citus) requer uma reinicialização do servidor (para selecionar a nova biblioteca compartilhada), seguida pelo comando ALTER EXTENSION UPDATE. A falha na execução de ambas as etapas pode causar erros ou falhas. A Hiperescala (Citus), portanto, valida se há correspondência entre a versão do código e da extensão, e gera erros se não corresponderem '.

Esse valor é válido para o coordenador e o padrão é true. Em casos raros, processos complexos de atualização podem exigir a definição desse parâmetro como false, desabilitando assim a verificação.

citus.log_distributed_deadlock_detection (booliano)

Se é preciso registrar o processamento relacionado à detecção de deadlock distribuído no registro do servidor. O padrão é false.

citus.distributed_deadlock_detection_factor (número de ponto flutuante)

Define o tempo de espera antes de verificar se há deadlocks distribuídos. Especificamente, o tempo de espera será esse valor multiplicado pela configuração do PostgreSQL' [deadlock_timeout](#). O valor padrão é [2](#). Um valor [-1](#) desabilita a detecção de deadlock distribuído.

citus.node_connection_timeout (inteiro)

A [citus.node_connection_timeout](#) GUC define a duração máxima (em milissegundos) de espera para o estabelecimento da conexão. A Hiperescala (Citus) gerará um erro se o tempo limite expirar antes de, no

mínimo, uma conexão de trabalho ser estabelecida. Essa GUC afeta as conexões do coordenador para os trabalhos — e entre eles.

- Padrão: cinco segundos
- Mínimo: 10 milissegundos
- Máximo: uma hora

```
-- set to 30 seconds
ALTER DATABASE foo
SET citus.node_connection_timeout = 30000;
```

Estatísticas de Consulta

citus.stat_statements_purge_interval (inteiro)

Define a frequência na qual o maintenance daemon remove os registros de [citus_stat_statements](#) que não correspondem no `pg_stat_statements`. Esse valor de configuração define o intervalo de tempo entre as limpezas em segundos, com um valor padrão de 10. Um valor de 0 desabilita as limpezas.

```
SET citus.stat_statements_purge_interval T0 5;
```

Esse parâmetro é válido para o coordenador e pode ser alterado no tempo de execução.

Carregamento de dados

citus.multi_shard_commit_protocol (enumeração)

Define o protocolo de confirmação a ser usado ao executar COPY em uma tabela distribuída por hash. Em cada posicionamento de fragmento individual, o comando COPY é executado em um bloco de transação para garantir que nenhum dado seja ingerido se ocorrer um erro durante o COPY. No entanto, há um caso de falha específico em que o COPY é realizado com êxito em todos os posicionamentos, mas ocorre uma falha (de hardware) antes que todas as transações sejam confirmadas. Nesse caso, o parâmetro pode ser usado para prevenir a perda de dados, optando entre os seguintes protocolos de confirmação:

- **2pc**: (padrão) as transações nas quais o COPY é executado nos posicionamentos de fragmentos são preparadas primeiro usando o ' [protocolo 2PC](#) do PostgreSQL e, em seguida, confirmadas. As confirmações com falha podem ser recuperadas ou anuladas manualmente usando COMMIT PREPARED ou ROLLBACK PREPARED, respectivamente. Ao usar 2pc, [as transações máximas preparadas](#) devem ser aumentadas em todos os trabalhos, normalmente para o mesmo valor que as `conexões_máximas`.
- **1pc**: as transações em que o COPY é executado nos posicionamentos de fragmentos são confirmadas em uma única vez. Os dados poderão ser perdidos se ocorrer uma falha na confirmação depois que o COPY for realizado com êxito em todos os posicionamentos (raro).

citus.shard_replication_factor (inteiro)

Define o fator de replicação para os fragmentos, ou seja, o número de nós nos quais os fragmentos serão colocados. O padrão é 1. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução. O valor ideal para esse parâmetro depende do tamanho do grupo e da taxa de falha do nó. Por exemplo, se você executar grupos grandes, pode querer aumentar esse fator de replicação e observar falhas do nó com mais frequência.

citus.shard_count (inteiro)

Define a contagem de fragmentos para as tabelas particionadas por hash e o padrão é 32. Esse valor é usado pelo UDF [create_distributed_table](#) ao criar tabelas particionadas por hash. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

citus.shard_max_size (inteiro)

Define o tamanho máximo do fragmento antes de ser dividido. O padrão é 1 GB. Quando o tamanho do arquivo de origem ' (usado no processo de preparo) para um fragmento exceder esse valor de configuração, o banco de

dados criará um novo fragmento. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

Configuração do planejador

`citus.limit_clause_row_fetch_count` (inteiro)

Define o número de linhas a serem buscadas por tarefa para uma otimização da cláusula de limite. Em alguns casos, as consultas selecionadas com cláusulas de limite podem precisar buscar em todas as linhas de cada tarefa para gerar resultados. Nesses cenários, e onde uma aproximação produziria resultados significativos, esse valor de configuração define o número de linhas a serem buscadas de cada fragmento. Por padrão, as aproximações de limite são desabilitadas e esse parâmetro é definido como 1. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.count_distinct_error_rate` (número de ponto flutuante)

A Hiperescala (Citus) pode calcular aproximações de `count(distinct)` usando a extensão `postgresql-hll`. Essa entrada de configuração define a taxa de erro desejada ao calcular a `count(distinct)`. O padrão 0,0 desabilita as aproximações da `count(distinct)`; e 1,0 não fornece nenhuma garantia sobre a precisão dos resultados. É recomendável definir esse parâmetro como 0,005 para obter melhores resultados. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.task_assignment_policy` (enumeração)

NOTE

Esse GUC é aplicável somente quando o `shard_replication_factor` for maior que um, ou para consultas nas `reference_tables`.

Define a política a ser usada ao atribuir tarefas aos trabalhos. O coordenador atribui tarefas aos trabalhos com base nos locais do fragmento. Esse valor de configuração especifica a política a ser usada ao fazer essas atribuições. Atualmente, há três políticas de atribuição de tarefas possíveis que podem ser usadas.

- **greedy**: a política de greedy é o padrão e visa distribuir uniformemente as tarefas entre os trabalhos.
- **round-robin**: a política de round-robin atribui tarefas aos trabalhos em rodízio, alternando entre réplicas diferentes. Essa política permite uma melhor utilização do grupo quando a contagem de fragmentos de uma tabela for baixa, comparada ao número de trabalhos.
- **first-replica**: A política de first-replica atribui tarefas com base na ordem de inserção dos posicionamentos (réplicas) para os fragmentos. Em outras palavras, a consulta de um fragmento é atribuída ao trabalho que tem a primeira réplica desse fragmento. Esse método permite que você tenha garantias consistentes sobre quais fragmentos serão usados em quais nós (ou seja, garantias de residência de memória mais eficientes).

Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

Transferência de dados intermediários

`citus.binary_worker_copy_format` (booliano)

Use o formato binário de cópia para transferir dados intermediários entre os trabalhos. Nas uniões de tabelas grandes, a Hiperescala (Citus) pode ter que reparticionar e colocar os dados em ordem aleatória dinamicamente entre diferentes trabalhos. Por padrão, esses dados são transferidos em formato de texto. Habilite esse parâmetro instrui o banco de dados a usar o formato binário de serialização do PostgreSQL para transferir esses dados. Esse parâmetro é válido para os trabalhos e precisa ser alterado no arquivo `postgresql.conf`. Depois de editar o arquivo de configuração, os usuários podem enviar um sinal `SIGHUP` ou reiniciar o servidor para que essa alteração entre em vigor.

`citus.binary_master_copy_format` (booliano)

Use o formato binário de cópia para transferir dados entre o coordenador e os trabalhos. Ao executar consultas distribuídas, os trabalhos transferem seus resultados intermediários para o coordenador para agregação final. Por padrão, esses dados são transferidos em formato de texto. Habilite esse parâmetro instrui o banco de

dados a usar o formato binário de serialização do PostgreSQL para transferir esses dados. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.max_intermediate_result_size` (íntero)

O tamanho máximo em KB dos resultados intermediários para CTEs que não podem ser enviados para nós de trabalho para execução e para subconsultas complexas. O padrão é 1 GB. Um valor de 1 significa sem limite. As consultas que excederem o limite serão canceladas e gerarão uma mensagem de erro.

DDL

`citus.enable_ddl_propagation` (booleano)

Especifica se as alterações de DDL devem ser propagadas automaticamente do coordenador para todos os trabalhos. O valor padrão é true. Como algumas alterações de esquema exigem um bloqueio exclusivo de acesso a tabelas — e como a propagação automática se aplica a todos os trabalhos sequencialmente — um grupo da Hiperescala (Citus) pode ficar temporariamente menos responsivo. Você pode optar por desabilitar essa configuração e propagar as alterações manualmente.

Configuração do executor

Geral

`citus.all_modifications_commutative`

A Hiperescala (Citus) impõe regras de comutatividade e obtém bloqueios apropriados para modificar operações, a fim de garantir a correção de comportamento. Por exemplo, ela pressupõe que uma instrução INSERT faz comutação com outra instrução INSERT, mas não com uma instrução UPDATE ou DELETE. Da mesma forma, ela pressupõe que uma instrução UPDATE ou DELETE não faça comutação com outra instrução UPDATE ou DELETE. Essa precaução significa que as instruções UPDATE e DELETE exigem que a Hiperescala (Citus) obtenha bloqueios mais fortes.

Se você tiver instruções UPDATE que estejam comutadas com instruções INSERT ou outras UPDATE, defina esse parâmetro como true. Quando esse parâmetro for definido como true, todos os comandos serão considerados comutativos e exigirão um bloqueio compartilhado, o que pode melhorar a produtividade geral. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.remote_task_check_interval` (íntero)

Define a frequência na qual a Hiperescala (Citus) verifica os status dos trabalhos gerenciados pelo executor do rastreador de tarefas. O padrão é 10 ms. O coordenador atribui tarefas aos trabalhos e, em seguida, verifica regularmente com eles sobre o progresso de cada tarefa¹. Esse valor de configuração define o intervalo de tempo entre duas verificações subsequentes. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.task_executor_type` (enumeração)

A Hiperescala (Citus) tem três tipos de executores para realizar consultas de SELECT distribuídas. O executor desejado pode ser selecionado, definindo este parâmetro de configuração. Os valores aceitáveis para esse parâmetro são:

- **adaptativo**: o padrão. É ideal para respostas rápidas a consultas que envolvem agregações e uniões com localização compartilhada que contêm vários fragmentos.
- **rastreador de tarefas**: o executor do rastreador de tarefas é adequado para consultas complexas e execução prolongada, que exigem embaralhamento de dados entre os nós de trabalho e gerenciamento eficiente de recursos.
- **tempo real**: (obsoleto) tem uma finalidade semelhante à do executor adaptável, mas é menos flexível e pode causar mais pressão de conexão nos nós de trabalho.

Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.multi_task_query_log_level` (enumeração) {#multi_task_logging}

Define um nível de registro para qualquer consulta que gere mais de uma tarefa (ou seja, que atinge mais de um fragmento). O registro em log é útil durante uma migração de aplicativo multilocatário, pois você pode escolher um erro ou aviso para essas consultas para encontrá-las e adicionar um filtro de ID de _locatário nelas.

Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução. O valor padrão para este parâmetro é 'off'.

Os valores suportados para esta enumeração são:

- **desativar**: desativa o registro de log de todas as consultas que geram várias tarefas (ou seja, contêm vários fragmentos)
- **depurar**: registra a instrução no nível de gravidade DEBUG.
- **registrar**: registra a instrução no nível de gravidade LOG. A linha de registro incluirá a consulta SQL que foi executada.
- **aviso**: registra a instrução no nível de gravidade NOTICE.
- **aviso**: registra a instrução no nível de gravidade WARNING.
- **erro**: registra a instrução no nível de gravidade ERROR.

Pode ser útil usar `error` durante o teste de desenvolvimento e um nível de registro inferior, como `log`, durante a implantação de produção real. Escolher `log` fará com que as consultas de várias tarefas apareçam nos registros do banco de dados com a própria consulta mostrada após "STATEMENT."

```
LOG: multi-task query about to be executed
HINT: Queries are split to multiple tasks if they have to be split into several queries on the workers.
STATEMENT: select * from foo;
```

`citus.enable_repartition_joins` (booleano)

Normalmente, ocorrerá uma falha com mensagem de erro na tentativa de executar uniões de repartição com o executor adaptável. No entanto, definir `citus.enable_repartition_joins` como true permite que a Hiperescala (Citus) alterne temporariamente para o executor do rastreador de tarefas para executar a união. O valor padrão é false.

Configuração do executor do rastreador de tarefas

`citus.task_tracker_delay` (inteiro)

Esse parâmetro define o tempo de suspensão do rastreador de tarefas entre os ciclos do gerenciamento de tarefas. O padrão é 200 ms. O processo do rastreador de tarefas é ativado regularmente, percorre todas as tarefas atribuídas a ele e agenda e executa essas tarefas. Em seguida, o rastreador de tarefas é suspenso por um período antes de percorrer essas tarefas novamente. Esse valor de configuração determina a duração desse período de suspensão. Esse parâmetro é válido para os trabalhos e precisa ser alterado no arquivo postgresql.conf. Depois de editar o arquivo de configuração, os usuários podem enviar um sinal SIGHUP ou reiniciar o servidor para que a alteração entre em vigor.

Esse parâmetro pode ser diminuído para cortar o atraso causado pelo executor do rastreador de tarefas, reduzindo o tempo entre os ciclos de gerenciamento. A diminuição do atraso é útil em casos em que as consultas de fragmento são curtas e, portanto, atualizam seu status regularmente.

`citus.max_assign_task_batch_size` (inteiro)

O executor do rastreador de tarefas no coordenador atribui de maneira síncrona as tarefas em lotes ao daemon nos trabalhos. Esse parâmetro define o número máximo de tarefas a serem atribuídas em um único lote. Escolher um tamanho de lote maior permite uma atribuição de tarefa mais rápida. No entanto, se o número de trabalhos for grande, poderá levar mais tempo para que todos os trabalhos obtenham tarefas. Esse parâmetro é válido para o coordenador e pode ser definido no tempo de execução.

`citus.max_running_tasks_per_node` (inteiro)

O processo do rastreador de tarefas agenda e executa as tarefas atribuídas a ele, conforme apropriado. Esse valor de configuração define o número máximo de tarefas a serem executadas simultaneamente em um nó em um tempo específico. O padrão é 8.

O limite garante que você não tenha muitas tarefas ocorrendo no disco ao mesmo tempo e ajuda a evitar a contenção de E/S do disco. Se as consultas forem alimentadas por meio da memória ou de SSDs, você poderá

aumentar o máximo de_tarefas_em execução_por_nó sem muitos problemas.

citus.partition_buffer_size (inteiro)

Define o tamanho do buffer a ser usado para operações de partição. O padrão é 8 MB. A Hiperescala (Citus) permite que os dados da tabela sejam reparticionados em vários arquivos ao unir duas tabelas grandes. Depois que esse buffer de partição for preenchido, os dados reparticionados serão liberados para arquivos no disco. Essa entrada de configuração é válida para os trabalhos e pode ser definida no tempo de execução.

Saída EXPLAIN

citus.explain_all_tasks (booleano)

Por padrão, a Hiperescala (Citus) mostra a saída de uma única tarefa arbitrária ao executar **EXPLAIN** em uma consulta distribuída. Na maioria dos casos, a saída EXPLAIN será semelhante entre as tarefas. Ocasionalmente, algumas das tarefas serão planejadas de maneira diferente ou terão tempos de execução muito maiores. Nesses casos, pode ser útil habilitar esse parâmetro; em seguida, a saída EXPLAIN incluirá todas as tarefas. A explicação de todas as tarefas pode fazer com que o EXPLAIN demore mais.

Parâmetros PostgreSQL

- [DateStyle](#) - Define o formato de exibição dos valores de data e hora.
- [IntervalStyle](#) - Define o formato de exibição para valores de intervalo.
- [TimeZone](#) - Define o fuso horário para exibir e interpretar carimbos de hora.
- [application_name](#) - Define o nome do aplicativo a ser informado nas estatísticas e registros.
- [array_nulls](#) - Ativa a entrada de elementos NULL em matrizes.
- [autovacuum](#) - Inicia o subprocesso autovacuum.
- [autovacuum_analyze_scale_factor](#) - Número de inserções, atualizações ou exclusões de tupla antes de analisar como uma fração de reltuples.
- [autovacuum_analyze_threshold](#) - Número mínimo de inserções, atualizações ou exclusões de tupla antes da análise.
- [autovacuum_naptime](#) - Tempo de suspensão entre as execuções de autovacuum.
- [autovacuum_vacuum_cost_delay](#) - Atraso do custo do vacuum em milissegundos para autovacuum.
- [autovacuum_vacuum_cost_limit](#) - Valor de custo do vacuum disponível antes da suspensão para autovacuum.
- [autovacuum_vacuum_scale_factor](#) - Número de atualizações ou exclusões de tupla antes do vacuum como uma fração de reltuples.
- [autovacuum_vacuum_threshold](#) - Número mínimo de atualizações ou exclusões de tupla antes do vacuum.
- [autovacuum_work_mem](#) - Define a memória máxima a ser usada por cada processo de trabalho de autovacuum.
- [backend_flush_after](#) - Número de páginas após as quais as gravações executadas anteriormente são liberadas para o disco.
- [backslash_quote](#) - Define se "" é permitido em literais de cadeia de caracteres.
- [bgwriter_delay](#) - Tempo de suspensão da gravação em segundo plano entre os ciclos.
- [bgwriter_flush_after](#) - Número de páginas após as quais as gravações executadas anteriormente são liberadas para o disco.
- [bgwriter_lru_maxpages](#) - Número máximo de páginas de LRU do gravador em segundo plano a serem liberadas por ciclo.
- [bgwriter_lru_multiplier](#) - Múltiplo do uso médio do buffer a ser liberado por ciclo.
- [bytea_output](#) - Define o formato de saída para bytes.
- [check_function_bodies](#) - Verifica os corpos da função durante CREATE FUNCTION.
- [checkpoint_completion_target](#) - Tempo gasto na limpeza de buffers sujos durante o ponto de verificação, como uma fração do intervalo do ponto de verificação.
- [checkpoint_timeout](#) - Define o tempo máximo entre os pontos de verificação de WAL automáticos.

- `checkpoint_warning` - Ativa os avisos se os segmentos do ponto de verificação forem preenchidos com mais frequência do que isso.
- `client_encoding` - Define a codificação do conjunto de caracteres do cliente.
- `client_min_messages` - Define os níveis de mensagem que são enviados ao cliente.
- `commit_delay` - Define o atraso em microssegundos entre a confirmação da transação e a liberação do WAL para o disco.
- `commit_siblings` - Define o mínimo de transações abertas simultâneas antes de executar `commit_delay`.
- `constraint_exclusion` - Permite que o planejador use restrições para otimizar as consultas.
- `cpu_index_tuple_cost` - Define a estimativa do planejador para o custo de processamento de cada entrada do índice durante uma verificação de índice.
- `cpu_operator_cost` - Define a estimativa do planejador para o custo de processamento de cada operador ou chamada de função.
- `cpu_tuple_cost` - Define a estimativa do planejador para o custo de processamento de cada tupla (linha).
- `cursor_tuple_fraction` - Define a estimativa da fração do planejador para as linhas de um cursor que serão recuperadas.
- `deadlock_timeout` - Define o tempo, em milissegundos, para aguardar um bloqueio antes de verificar se há deadlock.
- `debug_pretty_print` - Os recuos analisam e planejam as exibições de árvore.
- `debug_print_parse` - Registra a árvore de análise de cada consulta.
- `debug_print_plan` - Registra o plano de execução de cada consulta.
- `debug_print_rewritten` - Registra a árvore de análise reescrita de cada consulta.
- `default_statistics_target` - Define o destino das estatísticas padrão.
- `default_tablespace` - Define o espaço de tabela padrão nos quais criar tabelas e índices.
- `default_text_search_config` - Define a configuração de pesquisa de texto padrão.
- `default_transaction_deferrable` - Define o status padrão adiável de novas transações.
- `default_transaction_isolation` - Define o nível de isolamento da transação de cada nova transação.
- `default_transaction_read_only` - Define o status somente leitura padrão de novas transações.
- `default_with_oids` - Cria, por padrão, novas tabelas com OIDs.
- `effective_cache_size` - Define a suposição do planejador sobre o tamanho do cache de disco.
- `enable_bitmapscan` - Permite o uso do planejador de planos de verificação de bitmap.
- `enable_gathermerge` - Permite o uso do planejador de planos de mesclagem de coletas.
- `enable_hashagg` - Permite o uso do planejador de planos de agregação em hash.
- `enable_hashjoin` - Permite o uso do planejador de planos de união por hash.
- `enable_indexonlyscan` - Permite o uso do planejador de planos de verificação somente de índice.
- `enable_indexscan` - Permite o uso do planejador de planos de verificação de índice.
- `enable_material` - Permite o uso do planejador para materialização.
- `enable_mergejoin` - Permite o uso do planejador de planos de união de mesclagens.
- `enable_nestloop` - Permite o uso do planejador de planos de união de loops aninhados.
- `enable_seqscan` - Permite o uso do planejador de planos de verificações sequenciais.
- `enable_sort` - Permite o uso do planejador de etapas de classificação explícitas.
- `enable_tidscan` - Permite o uso do planejador de planos de verificação de TID.
- `escape_string_warning` - Avisa sobre os escapes de barra invertida em literais de cadeia de caracteres comuns.
- `exit_on_error` - Encerra a sessão quando ocorre qualquer erro.
- `extra_float_digits` - Define o número de dígitos exibidos para valores de ponto flutuante.
- `force_parallel_mode` - Força o uso de recursos de consulta paralela.
- `fromCollapse_limit` - Define o tamanho da lista FROM, acima dele, as subconsultas não serão recolhidas.

- `geqo` - Ativa a otimização de consulta herdada.
- `geqo_effort` - GEQO: esforço é usado para definir o padrão para outros parâmetros GEQO.
- `geqo_generations` - GEQO: número de iterações do algoritmo.
- `geqo_pool_size` - GEQO: número de indivíduos na população.
- `geqo_seed` - GEQO: semente para seleção de caminho aleatório.
- `geqo_selection_bias` - GEQO: pressão seletiva na população.
- `geqo_threshold` - Define o limite de itens FROM, acima dele, o GEQO será usado.
- `gin_fuzzy_search_limit` - Define o resultado máximo permitido para a pesquisa exata por GIN.
- `gin_pending_list_limit` - Define o tamanho máximo da lista pendente para o índice GIN.
- `idle_in_transaction_session_timeout` - Define a duração máxima permitida de qualquer transação inativa.
- `joinCollapse_limit` - Define o tamanho da lista FROM, acima dele, as construções JOIN não serão niveladas.
- `lc_monetary` - Define o local para formatar os valores monetários.
- `lc_numeric` - Define o local para formatar os números.
- `lo_compat_privileges` - Habilita o modo de compatibilidade com versões anteriores para verificações de privilégios em objetos grandes.
- `lock_timeout` - Define a duração máxima permitida (em milissegundos) de qualquer espera por um bloqueio. 0 desativa essa opção.
- `log_autovacuum_min_duration` - Define o tempo mínimo de execução, acima dele, as ações de autovacuum serão registradas.
- `log_checkpoints` - Registra cada ponto de verificação.
- `log_connections` - Registra cada conexão bem-sucedida.
- `log_destination` - Define o destino para a saída de registro do servidor.
- `log_disconnections` - Registra o final de uma sessão, incluindo a duração.
- `log_duration` - Registra a duração de cada instrução SQL concluída.
- `log_error_verbosity` - Define o detalhamento das mensagens registradas.
- `log_lock_waits` - Registra longas esperas de bloqueio.
- `log_min_duration_statement` - Define o tempo mínimo de execução (em milissegundos), acima dele, as instruções serão registradas. -1 desabilita as durações da instrução de registro em log.
- `log_min_error_statement` - Faz com que todas as instruções que geram erros neste nível ou acima sejam registradas.
- `log_min_messages` - Define os níveis de mensagem que são registrados.
- `log_replication_commands` - Registra cada comando de replicação.
- `log_statement` - Define o tipo de instruções registradas.
- `log_statement_stats` - Para cada consulta, grava as estatísticas de desempenho acumuladas no registro do servidor.
- `log_temp_files` - Registra o uso de arquivos temporários maiores do que este número de kilobytes.
- `maintenance_work_mem` - Define a memória máxima a ser usada para operações de manutenção.
- `max_parallel_workers` - Define o número máximo de trabalhos paralelos que podem estar ativos ao mesmo tempo.
- `max_parallel_workers_per_gather` - Define o número máximo de processos paralelos por nó executor.
- `max_pred_locks_per_page` - Define o número máximo de tuplas bloqueadas por predicado, por página.
- `max_pred_locks_per_relation` - Define o número máximo de páginas e tuplas, bloqueadas por predicado, por relação.
- `max_standby_archive_delay` - Define o atraso máximo antes de cancelar consultas, quando um servidor de espera ativa estiver processando dados de WAL arquivados.
- `max_standby_streaming_delay` - Define o atraso máximo antes de cancelar consultas, quando um servidor de espera ativa estiver processando dados de WAL transmitidos.

- `max_sync_workers_per_subscription` - Número máximo de trabalhos de sincronização de tabela por assinatura.
- `max_wal_size` - Define o tamanho do WAL que aciona um ponto de verificação.
- `min_parallel_index_scan_size` - Define a quantidade mínima de dados de índice para uma verificação paralela.
- `min_wal_size` - Define o tamanho mínimo para reduzir o WAL.
- `operator_precedence_warning` - Emite um aviso para construções que mudaram de significado desde o PostgreSQL 9.4.
- `parallel_setup_cost` - Define a estimativa do planejador para o custo de inicialização de processos de trabalho para uma consulta paralela.
- `parallel_tuple_cost` - Define a estimativa do planejador para o custo de encaminhamento de cada tupla (linha) do trabalho para o back-end mestre.
- `pg_stat_statements.save` - Salva estatísticas depg_stat_statements em desligamentos do servidor.
- `pg_stat_statements.track` - Seleciona quais instruções são rastreadas por pg_stat_statements.
- `pg_stat_statements.track_utility` - Seleciona se os comandos do utilitário serão rastreados por pg_stat_statements.
- `quote_all_identifiers` - Coloca entre aspas todos os identificadores ao gerar fragmentos de SQL.
- `random_page_cost` - Define a estimativa do planejador para o custo de uma página de disco buscada não sequencialmente.
- `row_security` - Habilita a segurança da linha.
- `search_path` - Define a ordem de pesquisa do esquema para nomes que não são qualificados pelo esquema.
- `seq_page_cost` - Define a estimativa do planejador para o custo de uma página de disco buscada sequencialmente.
- `session_replication_role` - Define o comportamento da sessão para gatilhos e regras de regravação.
- `standard_conforming_strings` - Faz com que as cadeias de caracteres '...' tratem literalmente as barras invertidas.
- `statement_timeout` - Define a duração máxima permitida (em milissegundos) de qualquer instrução. 0 desativa essa opção.
- `synchronize_seqscans` - Habilita verificações sequenciais sincronizadas.
- `synchronous_commit` - Define o nível de sincronização da transação atual.
- `tcp_keepalives_count` - Número máximo de retransmissões de keepalive do TCP.
- `tcp_keepalives_idle` - Tempo entre a emissão de keepalive do TCP.
- `tcp_keepalives_interval` - Tempo entre as retransmissões de manutenção do protocolo TCP.
- `temp_buffers` - Define o número máximo de buffers temporários usados por cada sessão do banco de dados.
- `temp_tablespaces` - Define os espaços de tabela a serem usados para tabelas temporárias e arquivos de classificação.
- `track_activities` - Coleta informações sobre a execução de comandos.
- `track_counts` - Coleta estatísticas sobre a atividade do banco de dados.
- `track_functions` - Coleta estatísticas de nível de função sobre a atividade do banco de dados.
- `track_io_timing` - Coleta estatísticas de tempo para a atividade de E/S do banco de dados.
- `transform_null_equals` - Trata "expr=NULL" como "expr IS NULL".
- `vacuum_cost_delay` - Atraso no custo de vacuum em milissegundos.
- `vacuum_cost_limit` - Valor do custo de vacuum disponível antes da suspensão.
- `vacuum_cost_page_dirty` - Custo de vacuum para uma página corrompida por vacuum.
- `vacuum_cost_page_hit` - Custo de vacuum para uma página encontrada no cache do buffer.
- `vacuum_cost_page_miss` - Custo de vacuum para uma página não encontrada no cache do buffer.
- `vacuum_defer_cleanup_age` - Número de transações em que a limpeza de VACUUM e HOT deve ser adiada, se houver.
- `vacuum_freeze_min_age` - Tempo mínimo em que o VACUUM deve congelar uma linha da tabela.

- [vacuum_freeze_table_age](#) - Tempo em que o VACUUM deve verificar a tabela inteira para congelar as tuplas.
- [vacuum_multixact_freeze_min_age](#) - Tempo mínimo em que o VACUUM deve congelar um MultiXactId em uma linha da tabela.
- [vacuum_multixact_freeze_table_age](#) - Tempo de multixact em que o VACUUM deve verificar a tabela inteira para congelar as tuplas.
- [wal_receiver_status_interval](#) - Define o intervalo máximo para o status do receptor WAL informar o primário.
- [wal_writer_delay](#) - Tempo entre as liberações de WAL realizadas no gravador WAL.
- [wal_writer_flush_after](#) - Quantidade de WAL registrado pelo gravador WAL que aciona uma liberação.
- [work_mem](#) - Define a quantidade de memória a ser usada por operações de classificação interna e tabelas de hash antes de gravar em arquivos de disco temporários.
- [xmlbinary](#) - Define como os valores binários devem ser codificados em XML.
- [xmloption](#) - Define se os dados XML em operações de análise e serialização implícitas devem ser considerados como documentos ou fragmentos de conteúdo.

Próximas etapas

- Outra forma de configuração, além dos parâmetros do servidor, são as [opções de configuração](#) do recurso em um grupo de servidores da Hiperescala (Citus).
- O banco de dados do PostgreSQL subjacente também tem [parâmetros de configuração](#).

Tabelas e exibições do sistema

21/05/2021 • 16 minutes to read

A Hiperescala (Citus) cria e mantém tabelas especiais que contêm informações sobre dados distribuídos no grupo de servidores. O nó coordenador consulta essas tabelas ao planejar como executar consultas nos nós de trabalho.

Metadados do coordenador

A Hiperescala (Citus) divide cada tabela distribuída em vários fragmentos lógicos com base na coluna de distribuição. O coordenador mantém as tabelas de metadados para acompanhar as estatísticas e as informações sobre a integridade e a localização desses fragmentos.

Nesta seção, descrevemos cada uma dessas tabelas de metadados e seu esquema. Você pode exibir e consultar essas tabelas usando SQL depois de fazer login no nó coordenador.

NOTE

Os grupos de servidores de Hiperescala (Citus) que executam versões mais antigas do mecanismo Citus podem não oferecer todas as tabelas listadas abaixo.

Tabela de partição

A tabela de partição pg_dist_ armazena metadados sobre quais tabelas no banco de dados são distribuídas. Para cada tabela distribuída, ela também armazena informações sobre o método de distribuição e informações detalhadas sobre a coluna de distribuição.

NOME	TIPO	DESCRIÇÃO
logicalrelid	regclass	Tabela distribuída à qual essa linha corresponde. Esse valor faz referência à coluna relfilenode na tabela do catálogo do sistema pg_class.
partmethod	char	O método usado para particionamento/distribuição. Os valores desta coluna correspondentes a métodos de distribuição diferentes são acrescentar: 'a', hash: 'h', tabela de referência: 'n'
partkey	text	Informações detalhadas sobre a coluna de distribuição, incluindo número de coluna, tipo e outras informações relevantes.
colocationid	Número inteiro	Grupo de colocação ao qual essa tabela pertence. As tabelas no mesmo grupo permitem junções colocadas e rollups distribuídos, entre outras otimizações. Esse valor faz referência à coluna colocationid na tabela pg_dist_colocation.

NOME	TIPO	DESCRIÇÃO
repmodel	char	O método usado para replicação de dados. Os valores desta coluna correspondentes a métodos de replicação diferentes são: replicação baseada em instrução Citus: 'c', replicação de streaming postgresql: 's', protocolo 2PC (para tabelas de referência): 't'

```

SELECT * from pg_dist_partition;
 logicalrelid | partmethod |
 | colocatoinid | repmodel
-----+-----+
-----+-----+
github_events | h      | {VAR :varno 1 :varattno 4 :vartype 20 :vartypmod -1 :varcollid 0 :varlevelsup
0 :varnoold 1 :varoattno 4 :location -1} |           2 | c
(1 row)

```

Tabela de fragmentos

A tabela de fragmentos pg_dist_armazena metadados sobre fragmentos individuais de uma tabela.

Pg_dist_shard tem informações sobre a quais fragmentos de tabela distribuída pertencem e estatísticas sobre a coluna de distribuição para fragmentos. Para acrescentar tabelas distribuídas, essas estatísticas correspondem aos valores mín./máx. da coluna de distribuição. Para tabelas hash distribuídas, elas são intervalos de token de hash atribuídos a esse fragmento. Essas estatísticas são usadas para remover fragmentos não relacionados durante consultas SELECT.

NOME	TIPO	DESCRIÇÃO
logicalrelid	regclass	Tabela distribuída à qual essa linha corresponde. Esse valor faz referência à coluna relfilenode na tabela do catálogo do sistema pg_class.
shardid	BIGINT	Identificador globalmente exclusivo atribuído a este fragmento.
shardstorage	char	Tipo de armazenamento usado para esse fragmento. Os diferentes tipos de armazenamento são discutidos na tabela a seguir.
shardminvalue	text	Para acrescentar tabelas distribuídas, o valor mínimo da coluna de distribuição neste fragmento (inclusivo). Para tabelas hash distribuídas, o valor de token de hash mínimo atribuído a esse fragmento (inclusivo).
shardmaxvalue	text	Para acrescentar tabelas distribuídas, o valor mínimo da coluna de distribuição neste fragmento (inclusivo). Para tabelas hash distribuídas, o valor de token de hash máximo atribuído a esse fragmento (inclusivo).

```

SELECT * from pg_dist_shard;
  logicalrelid | shardid | shardstorage | shardminvalue | shardmaxvalue
-----+-----+-----+-----+
github_events | 102026 | t      | 268435456   | 402653183
github_events | 102027 | t      | 402653184   | 536870911
github_events | 102028 | t      | 536870912   | 671088639
github_events | 102029 | t      | 671088640   | 805306367
(4 rows)

```

Tipos de armazenamento de fragmentos

A coluna shardstorage no fragmento pg_dist_ indica o tipo de armazenamento usado para o fragmento. Veja, abaixo, uma breve visão geral dos diferentes tipos de armazenamento de fragmento e sua representação.

TIPO DE ARMAZENAMENTO	VALOR DE SHARDSTORAGE	DESCRIÇÃO
TABLE	't'	Indica que o fragmento armazena dados pertencentes a uma tabela distribuída regular.
COLUMNAR	'c'	Indica que o fragmento armazena dados de coluna. (Usado por tabelas cstore_fdw distribuídas)
FOREIGN	'f'	Indica que o fragmento armazena dados externos. (Usado por tabelas file_fdw distribuídas)

Tabela de posicionamento de fragmento

A tabela de posicionamento pg_dist_ controla a localização das réplicas de fragmento em nós de trabalho. Cada réplica de um fragmento atribuído a um nó específico é chamada de posicionamento de fragmento. Esta tabela armazena informações sobre a integridade e a localização de cada posicionamento do fragmento.

NOME	TIPO	DESCRIÇÃO
shardid	BIGINT	Identificador de fragmento associado a este posicionamento. Esse valor faz referência à coluna shardid na tabela do catálogo pg_dist_shard.
shardstate	INT	Descreve o estado desse posicionamento. Os diferentes estados de fragmento são discutidos na seção abaixo.
shardlength	BIGINT	Para acrescentar tabelas distribuídas, o tamanho do posicionamento do fragmento no nó de trabalho em bytes. Para tabelas hash distribuídas, zero.
placementid	BIGINT	Identificador exclusivo gerado automaticamente para cada posicionamento individual.

NOME	TIPO	DESCRIÇÃO
groupid	INT	Denota um grupo de um servidor primário e zero ou mais servidores secundários quando o modelo de replicação de streaming é usado.

```

SELECT * from pg_dist_placement;
shardid | shardstate | shardlength | placementid | groupid
-----+-----+-----+-----+
102008 |      1 |      0 |      1 |      1
102008 |      1 |      0 |      2 |      2
102009 |      1 |      0 |      3 |      2
102009 |      1 |      0 |      4 |      3
102010 |      1 |      0 |      5 |      3
102010 |      1 |      0 |      6 |      4
102011 |      1 |      0 |      7 |      4

```

Estados de posicionamento do fragmento

A hiperescala (Citus) gerencia a integridade do fragmento em uma base por posicionamento. Se um posicionamento colocar o sistema em um estado inconsistente, o Citus o marcará automaticamente como indisponível. O estado de posicionamento é registrado na tabela pg_dist_shard_placement, dentro da coluna shardstate. Veja aqui uma breve visão geral dos diferentes estados de posicionamento do fragmento:

NOME DO ESTADO	VALOR DE SHARDSTATE	DESCRIÇÃO
FINALIZED	1	O estado em que os novos fragmentos são criados. Os posicionamentos de fragmentos nesse estado são considerados atualizados e usados no planejamento e na execução de consultas.
INACTIVE	3	Os posicionamentos de fragmentos nesse estado são considerados inativos porque estão fora de sincronia com outras réplicas do mesmo fragmento. O estado pode ocorrer quando uma operação de acréscimo, modificação (INSERÇÃO, ATUALIZAÇÃO, EXCLUSÃO) ou DDL falha para esse posicionamento. O planejador de consulta ignorará posicionamentos nesse estado durante o planejamento e a execução. Os usuários podem sincronizar os dados nesses fragmentos com uma réplica finalizada como uma atividade em segundo plano.
TO_DELETE	4	Se o Citus tentar remover um posicionamento de fragmento em resposta a uma chamada de master_apply_delete_command e falhar, o posicionamento será movido para esse estado. Os usuários podem excluir esses fragmentos como uma atividade em segundo plano subsequente.

Tabela de nó de trabalho

A tabela de nó pg_dist_node contém informações sobre os nós de trabalho no cluster.

NOME	TIPO	DESCRÍÇÃO
nodeid	INT	Identificador gerado automaticamente para um nó individual.
groupid	INT	Identificador usado para denotar um grupo de um servidor primário e zero ou mais servidores secundários quando o modelo de replicação de streaming é usado. Por padrão, é o mesmo que o nodeid.
nodename	text	Nome do host ou endereço IP do nó de trabalho do PostgreSQL.
nodeport	INT	Número da porta na qual o nó de trabalho do PostgreSQL está escutando.
noderack	text	(Opcional) Informações de posicionamento do rack para o nó de trabalho.
hasmetadata	booleano	Reservado para uso interno.
isactive	booleano	Se o nó está ativo aceitando posicionamentos de fragmento.
noderole	text	Se o nó é um primário ou secundário
nodecluster	text	O nome do cluster que contém este nó
shouldhaveshards	booleano	Se for falso, os fragmentos serão movidos para fora do nó (drenados) durante o rebalanceamento, nem os fragmentos de novas tabelas distribuídas serão colocados no nó, a menos que estejam colocados com fragmentos já existentes

```
SELECT * from pg_dist_node;
 nodeid | groupid | nodename | nodeport | noderack | hasmetadata | isactive | noderole | nodecluster |
-----+-----+-----+-----+-----+-----+-----+-----+
 1 |      1 | localhost |    12345 | default  | f          | t        | primary   | default    | t
 2 |      2 | localhost |    12346 | default  | f          | t        | primary   | default    | t
 3 |      3 | localhost |    12347 | default  | f          | t        | primary   | default    | t
(3 rows)
```

Tabela de objeto distribuído

A tabela de objeto citus.pg_dist_object contém uma lista de objetos, como tipos e funções que foram criados no nó coordenador e propagados para nós de trabalho. Quando um administrador adiciona novos nós de trabalho ao

cluster, a Hiperescala (Citus) cria automaticamente cópias dos objetos distribuídos nos novos nós (na ordem correta para satisfazer as dependências de objeto).

NOME	TIPO	DESCRIÇÃO
classid	oid	Classe do objeto distribuído
objid	oid	ID do objeto distribuído
objsubid	Número inteiro	Subid do objeto distribuído, por exemplo, attnum
type	text	Parte do endereço estável usado durante as atualizações de pg
object_names	text[]	Parte do endereço estável usado durante as atualizações de pg
object_args	text[]	Parte do endereço estável usado durante as atualizações de pg
distribution_argument_index	Número inteiro	Válido somente para funções/procedimentos distribuídos
colocationid	Número inteiro	Válido somente para funções/procedimentos distribuídos

Os "endereços estáveis" identificam exclusivamente objetos de forma independente de um servidor específico. A Hiperescala (Citus) rastreia objetos durante uma atualização do PostgreSQL usando endereços estáveis criados com a função [pg_identify_object_as_address\(\)](#).

Veja aqui um exemplo de como `create_distributed_function()` adiciona entradas à tabela

```
citus.pg_dist_object :
```

```

CREATE TYPE stoplight AS enum ('green', 'yellow', 'red');

CREATE OR REPLACE FUNCTION intersection()
RETURNS stoplight AS $$
DECLARE
    color stoplight;
BEGIN
    SELECT *
        FROM unnest(enum_range(NULL::stoplight)) INTO color
        ORDER BY random() LIMIT 1;
    RETURN color;
END;
$$ LANGUAGE plpgsql VOLATILE;

SELECT create_distributed_function('intersection()');

-- will have two rows, one for the TYPE and one for the FUNCTION
TABLE citus.pg_dist_object;

```

```

-[ RECORD 1 ]-----+
classid          | 1247
objid            | 16780
objsubid         | 0
type              |
object_names      |
object_args       |
distribution_argument_index |
colocationid      |
-[ RECORD 2 ]-----+
classid          | 1255
objid            | 16788
objsubid         | 0
type              |
object_names      |
object_args       |
distribution_argument_index |
colocationid      |

```

Tabela de grupos de colocação

A tabela de colocação pg_dist_colocation contém informações sobre quais fragmentos de tabelas devem ser posicionados juntos ou [colocados](#). Quando duas tabelas estão no mesmo grupo de colocação, a Hiperescala (Citus) garante que os fragmentos com os mesmos valores de partição serão posicionados nos mesmos nós de trabalho. A colocação permite otimizações de junção, determinados rollups distribuídos e suporte de chave estrangeira. A colocação de fragmento é inferida quando as contagens de fragmentos, os fatores de replicação e os tipos de coluna de partição correspondem entre as duas tabelas; no entanto, um grupo de colocação personalizado pode ser especificado ao criar uma tabela distribuída, se desejado.

NOME	TIPO	DESCRIÇÃO
colocationid	INT	Identificador exclusivo para o grupo de colocação ao qual esta linha corresponde.
shardcount	INT	Contagem de fragmentos para todas as tabelas neste grupo de colocação
replicationfactor	INT	Fator de replicação para todas as tabelas neste grupo de colocação.
distributioncolumntype	oid	O tipo da coluna de distribuição para todas as tabelas neste grupo de colocação.

```

SELECT * from pg_dist_colocation;
+-----+-----+-----+-----+
| 2 |     32 |        2 |      20 |
+-----+-----+-----+-----+
(1 row)

```

Tabela de estratégias de rebalanceador

Esta tabela define estratégias que [rebalance_table_shards](#) podem usar para determinar onde mover fragmentos.

NOME	TIPO	DESCRIÇÃO
------	------	-----------

NOME	TIPO	DESCRIÇÃO
default_strategy	booleano	Se rebalance_table_shards deve escolher essa estratégia por padrão. Use citus_set_default_rebalance_strategy para atualizar esta coluna
shard_cost_function	regproc	Identificador para uma função de custo, que deve usar um shardid como bigint e retornar sua noção de um custo, como o tipo real
node_capacity_function	regproc	Identificador para uma função de capacidade, que deve usar um nodeid como int e retornar sua noção de capacidade de nó como tipo real
shard_allowed_on_node_function	regproc	Identificador para uma função que determinado shardid bigint, e nodeidarg int, retorna booleano para saber se Citus pode armazenar o fragmento no nó
default_threshold	float4	Límite para considerar um nó muito cheio ou muito vazio, que determina quando o rebalance_table_shards deve tentar mover os fragmentos
minimum_threshold	float4	Uma proteção para impedir que o argumento de limite de rebalance_table_shards() seja definido muito baixo

Uma instalação de Hiperescala (Citus) é fornecida com estas estratégias na tabela:

```
SELECT * FROM pg_dist_rebalance_strategy;
```

```
-[ RECORD 1 ]-----+
Name          | by_shard_count
default_strategy | true
shard_cost_function | citus_shard_cost_1
node_capacity_function | citus_node_capacity_1
shard_allowed_on_node_function | citus_shard_allowed_on_node_true
default_threshold | 0
minimum_threshold | 0
-[ RECORD 2 ]-----+
Name          | by_disk_size
default_strategy | false
shard_cost_function | citus_shard_cost_by_disk_size
node_capacity_function | citus_node_capacity_1
shard_allowed_on_node_function | citus_shard_allowed_on_node_true
default_threshold | 0.1
minimum_threshold | 0.01
```

A estratégia padrão, `by_shard_count`, atribui a cada fragmento o mesmo custo. Seu efeito é igualar a contagem de fragmentos entre nós. A outra estratégia predefinida, `by_disk_size`, atribui um custo a cada fragmento que corresponde a seu tamanho de disco em bytes, além dos fragmentos que estão colocados com ele. O tamanho

do disco é calculado usando `pg_total_relation_size`, portanto, inclui índices. Essa estratégia tenta atingir o mesmo espaço em disco em cada nó. Observe o limite de 0,1--ele impede a movimentação desnecessária de fragmentos causada por diferenças insignificantes no espaço em disco.

Criando estratégias de rebalanceador personalizado

Veja aqui exemplos de funções que podem ser usadas em novas estratégias de rebalanceador de fragmento e registradas no `pg_dist_rebalance_strategy` com a função `citus_add_rebalance_strategy`.

- Definindo uma exceção de capacidade de nó por padrão de nome do host:

```
CREATE FUNCTION v2_node_double_capacity(nodeidarg int)
RETURNS boolean AS $$ 
SELECT
    (CASE WHEN nodename LIKE '%.v2.worker.citusdata.com' THEN 2 ELSE 1 END)
FROM pg_dist_node where nodeid = nodeidarg
$$ LANGUAGE sql;
```

- Rebalanceamento por número de consultas que vão para um fragmento, conforme medido pelo `citus_stat_statements`:

```
-- example of shard_cost_function

CREATE FUNCTION cost_of_shard_by_number_of_queries(shardid bigint)
RETURNS real AS $$ 
SELECT coalesce(sum(calls)::real, 0.001) as shard_total_queries
FROM citus_stat_statements
WHERE partition_key is not null
    AND get_shard_id_for_distribution_column('tab', partition_key) = shardid;
$$ LANGUAGE sql;
```

- Isolando um fragmento específico (10000) em um nó (endereço '10.0.0.1'):

```
-- example of shard_allowed_on_node_function

CREATE FUNCTION isolate_shard_10000_on_10_0_0_1(shardid bigint, nodeidarg int)
RETURNS boolean AS $$ 
SELECT
    (CASE WHEN nodename = '10.0.0.1' THEN shardid = 10000 ELSE shardid != 10000 END)
FROM pg_dist_node where nodeid = nodeidarg
$$ LANGUAGE sql;

-- The next two definitions are recommended in combination with the above function.
-- This way the average utilization of nodes is not impacted by the isolated shard.
CREATE FUNCTION no_capacity_for_10_0_0_1(nodeidarg int)
RETURNS real AS $$ 
SELECT
    (CASE WHEN nodename = '10.0.0.1' THEN 0 ELSE 1 END)::real
FROM pg_dist_node where nodeid = nodeidarg
$$ LANGUAGE sql;
CREATE FUNCTION no_cost_for_10000(shardid bigint)
RETURNS real AS $$ 
SELECT
    (CASE WHEN shardid = 10000 THEN 0 ELSE 1 END)::real
$$ LANGUAGE sql;
```

Tabela de estatísticas de consultas

A Hiperscala (Citus) fornece `citus_stat_statements` para estatísticas sobre como as consultas estão sendo executadas e para quem. É análoga (e pode ser unida) à exibição de `pg_stat_statements` no PostgreSQL, que rastreia estatísticas sobre a velocidade da consulta.

Essa exibição pode rastrear consultas para locatários de origem em um aplicativo multilocatário, o que ajuda a decidir quando fazer o isolamento do locatário.

NOME	TIPO	DESCRIÇÃO
queryid	BIGINT	identificador (bom para junções de pg_stat_statements)
userid	oid	usuário que executou a consulta
dbid	oid	instância de banco de dados do coordenador
Consulta	text	cadeia de caracteres de consulta anônima
executor	text	Executor Citus usado: adaptável, em tempo real, rastreador de tarefas, roteador ou inserção-seleção
partition_key	text	valor da coluna de distribuição em consultas executadas pelo roteador, senão NULO
chamadas	BIGINT	número de vezes que a consulta foi executada

```
-- create and populate distributed table
create table foo ( id int );
select create_distributed_table('foo', 'id');
insert into foo select generate_series(1,100);

-- enable stats
-- pg_stat_statements must be in shared_preload_libraries
create extension pg_stat_statements;

select count(*) from foo;
select * from foo where id = 42;

select * from citus_stat_statements;
```

Resultados:

```

-[ RECORD 1 ]+-----
queryid      | -909556869173432820
userid       | 10
dbid         | 13340
query        | insert into foo select generate_series($1,$2)
executor     | insert-select
partition_key |
calls        | 1
-[ RECORD 2 ]+-----
queryid      | 3919808845681956665
userid       | 10
dbid         | 13340
query        | select count(*) from foo;
executor     | adaptive
partition_key |
calls        | 1
-[ RECORD 3 ]+-----
queryid      | 5351346905785208738
userid       | 10
dbid         | 13340
query        | select * from foo where id = $1
executor     | adaptive
partition_key | 42
calls        | 1

```

Restrições:

- Os dados de estatísticas não são replicados e não sobreviverão a falhas ou failover do banco de dados
- Rastreia um número limitado de consultas, definidas pelo GUC `pg_stat_statements.max` (padrão 5000)
- Para truncar a tabela, use a função `citus_stat_statements_reset()`

Atividade de consulta distribuída

A Hiperescala (Citus) fornece exibições especiais para inspecionar consultas e bloqueios em todo o cluster, incluindo consultas específicas de fragmentos usadas internamente para criar resultados para consultas distribuídas.

- `citus_dist_stat_activity`: mostra as consultas distribuídas que estão sendo executadas em todos os nós. Um superconjunto de `pg_stat_activity`, utilizável onde quer que esse último esteja.
- `citus_worker_stat_activity`: mostra consultas em trabalhos, incluindo consultas de fragmentos em fragmentos individuais.
- `citus_lock_waits`: consultas bloqueadas em todo o cluster.

As duas primeiras exibições incluem todas as colunas de `pg_stat_activity` além do host/porta do trabalho que iniciou a consulta e o host/porta do nó coordenador do cluster.

Por exemplo, considere a possibilidade de contar as linhas em uma tabela distribuída:

```

-- run from worker on localhost:9701

SELECT count(*) FROM users_table;

```

Podemos ver que a consulta aparece em `citus_dist_stat_activity`:

```

SELECT * FROM citus_dist_stat_activity;

-[ RECORD 1 ]-----+
query_hostname      | localhost
query_hostport     | 9701
master_query_host_name | localhost
master_query_host_port | 9701
transaction_number   | 1
transaction_stamp    | 2018-10-05 13:27:20.691907+03
datid               | 12630
datname              | postgres
pid                 | 23723
usesysid             | 10
username              | citus
application\_name    | psql
client\_addr          |
client\_hostname       |
client\_port            | -1
backend\_start         | 2018-10-05 13:27:14.419905+03
xact\_start            | 2018-10-05 13:27:16.362887+03
query\_start           | 2018-10-05 13:27:20.682452+03
state\_change          | 2018-10-05 13:27:20.896546+03
wait\_event_type       | Client
wait\_event             | ClientRead
state                  | idle in transaction
backend\_xid            |
backend\_xmin           |
query                 | SELECT count(*) FROM users_table;
backend\_type           | client backend

```

Essa consulta requer informações de todos os fragmentos. Algumas das informações estão no fragmento `users_table_102038`, que são armazenadas no `localhost:9700`. Podemos ver uma consulta acessando o fragmento, examinando a exibição `citus_worker_stat_activity`:

```

SELECT * FROM citus_worker_stat_activity;

-[ RECORD 1 ]-----
-----
query_hostname      | localhost
query_hostport     | 9700
master_query_host_name | localhost
master_query_host_port | 9701
transaction_number   | 1
transaction_stamp    | 2018-10-05 13:27:20.691907+03
datid               | 12630
datname              | postgres
pid                  | 23781
usesysid             | 10
username              | citus
application\_name    | citus
client\_addr          | ::1
client\_hostname       |
client\_port           | 51773
backend\_start         | 2018-10-05 13:27:20.75839+03
xact\_start            | 2018-10-05 13:27:20.84112+03
query\_start           | 2018-10-05 13:27:20.867446+03
state\_change          | 2018-10-05 13:27:20.869889+03
wait\_event_type       | Client
wait\_event             | ClientRead
state                 | idle in transaction
backend\_xid            |
backend\_xmin           |
query                 | COPY (SELECT count(*) AS count FROM users_table_102038 users_table WHERE true) TO
STDOUT
backend\_type           | client backend

```

O campo `query` mostra os dados sendo copiados do fragmento a ser contado.

NOTE

Se uma consulta de roteador (por exemplo, um único locatário em um aplicativo multilocatário, `SELECIONE

- DA tabela ONDE tenant_id = X` for executada sem um bloco de transação, as colunas_query_host_name e master_query_host_port serão NULAS em `citus_worker_stat_activity`.

Para ver como o `citus_lock_waits` funciona, podemos gerar uma situação de bloqueio manualmente. Primeiro, vamos configurar uma tabela de teste a partir do coordenador:

```

CREATE TABLE numbers AS
  SELECT i, 0 AS j FROM generate_series(1,10) AS i;
SELECT create_distributed_table('numbers', 'i');

```

Em seguida, usando duas sessões no coordenador, podemos executar esta sequência de instruções:

```

-- session 1                                -- session 2
-----
BEGIN;
UPDATE numbers SET j = 2 WHERE i = 1;
                                         BEGIN;
                                         UPDATE numbers SET j = 3 WHERE i = 1;
                                         -- (this blocks)

```

A exibição `citus_lock_waits` mostra a situação.

```
SELECT * FROM citus_lock_waits;

-[ RECORD 1 ]-----+-----+
waiting_pid          | 88624
blocking_pid          | 88615
blocked_statement     | UPDATE numbers SET j = 3 WHERE i = 1;
current_statement_in_blocking_process | UPDATE numbers SET j = 2 WHERE i = 1;
waiting_node_id       | 0
blocking_node_id      | 0
waiting_node_name     | coordinator_host
blocking_node_name    | coordinator_host
waiting_node_port      | 5432
blocking_node_port     | 5432
```

Neste exemplo, as consultas são provenientes do coordenador, mas a exibição também pode listar os bloqueios entre as consultas provenientes de trabalhos (executados com a Hiperescala (Citus) MX por exemplo).

Próximas etapas

- Saiba como algumas [funções de Hiperescala \(Citus\)](#) alteram tabelas do sistema
- Examine os conceitos de [nós e tabelas](#)

Definições internas do Azure Policy para o Banco de Dados do Azure para PostgreSQL

16/06/2021 • 5 minutes to read

Esta página é um índice de definições de políticas internas do [Azure Policy](#) para o Banco de Dados do Azure para PostgreSQL. Para obter políticas internas adicionais do Azure Policy para outros serviços, confira [Definições internas do Azure Policy](#).

O nome de cada definição de política interna leva à definição da política no portal do Azure. Use o link na coluna **Versão** para exibir a origem no [repositório GitHub do Azure Policy](#).

Banco de Dados do Azure para PostgreSQL

NOME (PORTAL DO AZURE)	DESCRIÇÃO	EFEITO(S)	VERSÃO (GITHUB)
Configurar a Proteção Avançada contra Ameaças como habilitada no banco de dados do Azure para servidores PostgreSQL	Habilite a Proteção Avançada contra Ameaças nos seus bancos de dados do Azure de nível Não Básico para que os servidores PostgreSQL detectem atividades anômalas que indiquem tentativas incomuns e possivelmente prejudiciais no sentido de acessar ou explorar bancos de dados.	DeployIfNotExists, desabilitado	1.0.0
A limitação de conexão deve estar habilitada para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a limitação de conexão habilitada. Essa configuração permite a otimização temporária da conexão por IP para muitas falhas inválidas de login de senha.	AuditIfNotExists, desabilitado	1.0.0
As desconexões devem ser registradas em log para os servidores de banco de dados PostgreSQL.	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem log_disconnections habilitada.	AuditIfNotExists, desabilitado	1.0.0

NOME	DESCRIÇÃO	EFEITO(S)	VERSÃO
'Impor conexão SSL' deve ser habilitada para servidores de banco de dados PostgreSQL	<p>O Banco de Dados do Azure para PostgreSQL dá suporte à conexão de seu servidor de Banco de Dados do Azure para PostgreSQL para aplicativos cliente usando o protocolo SSL. Impor conexões SSL entre seu servidor de banco de dados e os aplicativos cliente ajuda a proteger contra ataques de "intermediários" criptografando o fluxo de dados entre o servidor e seu aplicativo. Essa configuração impõe que o SSL sempre esteja habilitado para acessar seu servidor de banco de dados.</p>	Audit, desabilitado	1.0.1
O backup com redundância geográfica deve ser habilitado para o Banco de Dados do Azure para PostgreSQL	<p>O Banco de Dados do Azure para PostgreSQL permite que você escolha a opção de redundância para seu servidor de banco de dados. Ele pode ser definido como um armazenamento de backup com redundância geográfica no qual os dados não são armazenados apenas na região em que o servidor está hospedado, mas também é replicado para uma região emparelhada para dar a opção de recuperação em caso de falha de região. A configuração do armazenamento com redundância geográfica para backup só é permitida durante a criação do servidor.</p>	Audit, desabilitado	1.0.1

NOME	DESCRÍÇÃO	EFEITO(S)	VERSÃO
A criptografia de infraestrutura deve ser habilitada para servidores do Banco de Dados do Azure para PostgreSQL	Habilite a criptografia de infraestrutura para os servidores do Banco de Dados do Azure para PostgreSQL terem um nível mais alto de garantia de segurança dos dados. Quando a criptografia de infraestrutura está habilitada, os dados em repouso são criptografados duas vezes usando chaves gerenciadas da Microsoft em conformidade com o FIPS 140-2	Audit, Deny, desabilitado	1.0.0
Os pontos de verificação de log devem ser habilitados para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a configuração log_checkpoints habilitada.	AuditIfNotExists, desabilitado	1.0.0
As conexões de log devem ser habilitadas para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a configuração log_connections habilitada.	AuditIfNotExists, desabilitado	1.0.0
A duração do log deve ser habilitada para os servidores de banco de dados PostgreSQL	Esta política ajuda a auditar os bancos de dados PostgreSQL no ambiente sem a configuração log_duration habilitada.	AuditIfNotExists, desabilitado	1.0.0
O servidor PostgreSQL deve usar um ponto de extremidade de serviço de rede virtual	As regras de firewall baseadas em rede virtual são usadas para habilitar o tráfego de uma sub-rede específica para o Banco de Dados do Azure para PostgreSQL e ainda garantir que o tráfego permaneça dentro do limite do Azure. Essa política oferece uma forma de auditar se o Banco de Dados do Azure para PostgreSQL tem um ponto de extremidade de serviço de rede virtual em uso.	AuditIfNotExists, desabilitado	1.0.2

NOME	DESCRIÇÃO	EFEITO(S)	VERSÃO
Os servidores PostgreSQL devem usar chaves gerenciadas pelo cliente para criptografar dados inativos	<p>Use chaves gerenciadas pelo cliente para gerenciar a criptografia em repouso de seus servidores PostgreSQL. Por padrão, os dados são criptografados em repouso com chaves gerenciadas pelo serviço, mas as chaves gerenciadas pelo cliente normalmente são necessárias para atender aos padrões de conformidade regulatória. As chaves gerenciadas pelo cliente permitem que os dados sejam criptografados com uma chave do Azure Key Vault criada por você e de sua propriedade. Você tem total controle e responsabilidade pelo ciclo de vida da chave, incluindo rotação e gerenciamento.</p>	AuditIfNotExists, desabilitado	1.0.4
O ponto de extremidade privado deve ser habilitado para servidores PostgreSQL	<p>As conexões de ponto de extremidade privado impõem comunicações seguras habilitando a conectividade privada ao Banco de Dados do Azure para PostgreSQL. Configure uma conexão de ponto de extremidade privado para habilitar o acesso ao tráfego proveniente somente de redes conhecidas e impedir o acesso de todos os outros endereços IP, incluindo no Azure.</p>	AuditIfNotExists, desabilitado	1.0.2
O acesso à rede pública deve ser desabilitado para servidores flexíveis do PostgreSQL	<p>Desabilitar a propriedade de acesso à rede pública aprimora a segurança garantindo que o Banco de Dados do Azure para servidores flexíveis do PostgreSQL só possa ser acessado de um ponto de extremidade privado. Essa configuração desabilita estritamente o acesso de qualquer espaço de endereço público fora do intervalo de IP do Azure e nega todos os logons que correspondam a regras de firewall baseadas em IP ou em rede virtual.</p>	Audit, Deny, desabilitado	1.0.0

NOME	DESCRIÇÃO	EFEITO(S)	VERSAO
O acesso à rede pública deve ser desabilitado para servidores PostgreSQL	Desabilitar a propriedade de acesso à rede pública aprimora a segurança e garantir que o Banco de Dados do Azure para PostgreSQL só possa ser acessado de um ponto de extremidade privado. Essa configuração desabilita o acesso de qualquer espaço de endereço público fora do intervalo de IP do Azure e nega todos os logons que correspondam a regras de firewall baseadas em IP ou em rede virtual.	Audit, desabilitado	1.0.2

Próximas etapas

- Confira os internos no [repositório Azure Policy GitHub](#).
- Revise a [estrutura de definição do Azure Policy](#).
- Revisar [Compreendendo os efeitos da política](#).

Banco de dados do Azure para Vídeos PostgreSQL

21/05/2021 • 2 minutes to read

Esta página fornece conteúdos em vídeo para o aprendizado sobre o Banco de dados do Azure para PostgreSQL.

Visão geral: Banco de Dados do Azure para PostgreSQL e MySQL

[Abrir no Channel 9](#)

O Banco de Dados do Azure para PostgreSQL e o Banco de Dados do Azure para MySQL reúnem os mecanismos de banco de dados da Community Edition e as funcionalidades de um serviço totalmente gerenciado – assim, você pode se concentrar em seus aplicativos em vez de ter que gerenciar um banco de dados. Prepare-se para obter uma breve visão geral das vantagens de usar o serviço e veja algumas funcionalidades em ação.

Criar um servidor PostgreSQL ou MySQL

O Banco de Dados do Azure para PostgreSQL e para MySQL são serviços gerenciados que você usa para executar, gerenciar e escalar mecanismos de banco de dados do Community Edition altamente disponíveis na nuvem. Esses tutoriais em vídeo mostram como criar um servidor PostgreSQL ou MySQL em cerca de três minutos usando o portal do Azure. Caso você não tenha uma assinatura do Azure, [crie uma conta gratuita do Azure](#) antes de começar.

- [Tutorial em vídeo do PostgreSQL](#)
- [Tutorial em vídeo do MySQL](#)

Conheça com profundidade as funcionalidades de serviço gerenciado para MySQL e PostgreSQL

[Abrir no Channel 9](#)

O Banco de Dados do Azure para PostgreSQL e o Banco de Dados do Azure para MySQL reúnem os mecanismos de banco de dados da Community Edition e as funcionalidades de um serviço totalmente gerenciado. Prepare-se para aprofundar-se sobre o funcionamento desses serviços – como garantimos a alta disponibilidade e o rápido dimensionamento (dentro de segundos), para que possa atender às necessidades dos clientes. Você também conhecerá alguns investimentos subjacentes em segurança e disponibilidade em todo o mundo.

Desenvolver um aplicativo de análise inteligente com o PostgreSQL

[Abrir no Channel 9](#)

O Banco de Dados do Azure para PostgreSQL combina o mecanismo de banco de dados da Community Edition e as funcionalidades de um serviço totalmente gerenciado – assim, você pode se concentrar em seus aplicativos em vez de ter que gerenciar um banco de dados. Prepare-se para ver em ação como é fácil criar novas experiências, como adicionar Serviços cognitivos aos seus aplicativos devido à presença no Azure.

Como começar usar o novo Banco de dados do Azure para o serviço PostgreSQL

[Abrir no Channel 9](#)

Neste vídeo da Microsoft 2017 //Criar uma conferência, aprenda com dois antigos clientes a forma de usar o Banco de dados do Azure para o serviço PostgreSQL para obter uma inovação mais rápida. Aprenderemos como eles migraram para o serviço e discutiremos as próximas etapas no desenvolvimento do aplicativo deles. O vídeo aborda alguns dos principais recursos do serviço e discute as formas como você, como desenvolvedor, pode migrar os seus aplicativos já existentes ou desenvolver novos aplicativos que usem este serviço gerenciado de PostgreSQL no Azure.

Banco de dados do Azure para os parceiros de migração do PostgreSQL

21/05/2021 • 2 minutes to read

Para oferecer suporte amplo à sua solução do Banco de Dados do Azure para PostgreSQL, escolha entre uma ampla variedade de parceiros e ferramentas líderes do setor. Este artigo destaca as empresas parceiras da Microsoft com soluções de migração que dão suporte ao Banco de Dados do Azure para PostgreSQL.

Parceiros de migração

PARTNER (PARCEIRO)	DESCRÍÇÃO	LINKS	VÍDEOS
 snp technologies inc.	SNP Technologies A SNP Technologies é uma provedora de serviços somente na nuvem, criando soluções seguras e confiáveis para os negócios do futuro. A empresa acredita na geração de valor real para o seu negócio. Do pensamento à execução, a SNP Technologies compartilha um objetivo comum com os clientes, para transformar seu investimento em uma vantagem.	Site Twitter Contact	
 DB BEST TECHNOLOGIES	DB Best Technologies, LLC A DB Best ajuda os clientes a obterem a maior parte do serviço de banco de dados do Azure. A empresa oferece várias maneiras para começar, incluindo Projeto arquitetônico de estado futuro , Otimização de gerenciamento de dados para o Microsoft Data Platform , Serviços de Planejamento de Implantação do Microsoft Azure e Treinamento de Preparação para Plataforma de Dados do Azure .	Site Twitter YouTube Contact	

PARTNER (PARCEIRO)	DESCRÍÇÃO	LINKS	VÍDEOS
Pragmatic Works	<p>Pragmatic Works A Pragmatic Works é uma empresa de treinamento e consultoria com profundo conhecimento em gerenciamento e desempenho de dados, Business Intelligence, Big Data, Power BI e Azure. Com foco na otimização de dados e na melhoria da eficiência do Microsoft SQL Server e do gerenciamento de nuvem.</p>	Site Twitter YouTube Contact	
Infosys®	<p>Infosys A Infosys é líder global nos serviços e consultoria digitais mais recentes. Com mais de três décadas de experiência no gerenciamento de sistemas de empresas globais, a Infosys orienta habilmente os clientes por meio de sua jornada digital, permitindo as organizações com um núcleo alimentado por IA. Fazer isso ajuda a priorizar a execução de alteração. A Infosys também oferece às empresas uma escala digital ágil para oferecer níveis de desempenho sem precedentes e satisfação do cliente.</p>	Site Twitter YouTube Contact	
	<p>credativ a credativ é uma empresa independente de consultoria e serviços. Desde 1999, oferece serviços abrangentes e suporte técnico para a implementação e operação do software Open Source em aplicativos comerciais. Sua ampla gama de serviços inclui um consultoria estratégica, bom conselho técnico, treinamento qualificado e suporte personalizado até 24 horas por dia para todas as suas necessidades de TI.</p>	Marketplace Site Twitter YouTube Contact	

PARTNER (PARCEIRO)	DESCRÍÇÃO	LINKS	VÍDEOS
	<p>Pactera Pactera é uma empresa global que oferece serviços de consultoria, digital, tecnologia e operações para empresas líderes do mundo. De suas raízes em engenharia para a versão mais recente na transformação digital, oferece aos clientes uma vantagem competitiva. Suas ferramentas e metodologias comprovadas certificam que seus dados são seguro, autênticos e precisos.</p>	Site Twitter Contact	

Próximas etapas

Para saber mais sobre outros parceiros da Microsoft, consulte o site [Microsoft Partner](#).