How to Utilize DxOdyssey Tunnels

Accessing PostgreSQL on an Ubuntu Node

Introduction

Although it's been around for over 30 years, adoption rates—and use cases—for PostgreSQL are still growing. Its core distinctions of data correctness and integrity are particularly effective on Ubuntu nodes, which typify all of the innovation advantages of open source communities. PostgreSQL remains a viable choice for organizations across verticals not only because it's one of the few top 10 databases not backed by a specific vendor, but also because, as an object relational DBMS, it supports a variety of methods and data types for extremely extensible, scalable, real-time performant needs.

The Challenge

If not properly addressed, however, these same attributes can create cyber security nightmares for organizations, especially with the sort of distributed workloads at which PostgreSQL excels. Specifically, these security challenges include:

Multi-Tenancy and Multi-Tenant Clouds

» The horizontal scalability of PostgreSQL is perfect for multi-tenant software and multi-tenant cloud applications, especially when extended with Citus capabilities. Although multi-tenancy is becoming increasingly important in today's hybrid and multi-cloud world, it also increases the heterogeneity of enterprise computing environments and their cyber security demands. Horizontally scaling a single database (instead of deploying multiple ones) or sharding it reduces architectural complexity and associated costs, but shifts security concerns to fortifying transmissions between distributed settings. These environments may lack conventional perimeter defenses, many of which (such as VPNs and ACLs) expand attack surface and are tough to maintain, respectively. The more distributed transactional systems are for OLTP, for instance, the more distributed cyber security mechanisms have to be since intrusion opportunities increase with the number of points to defend.

Decentralized Application Add-Ons

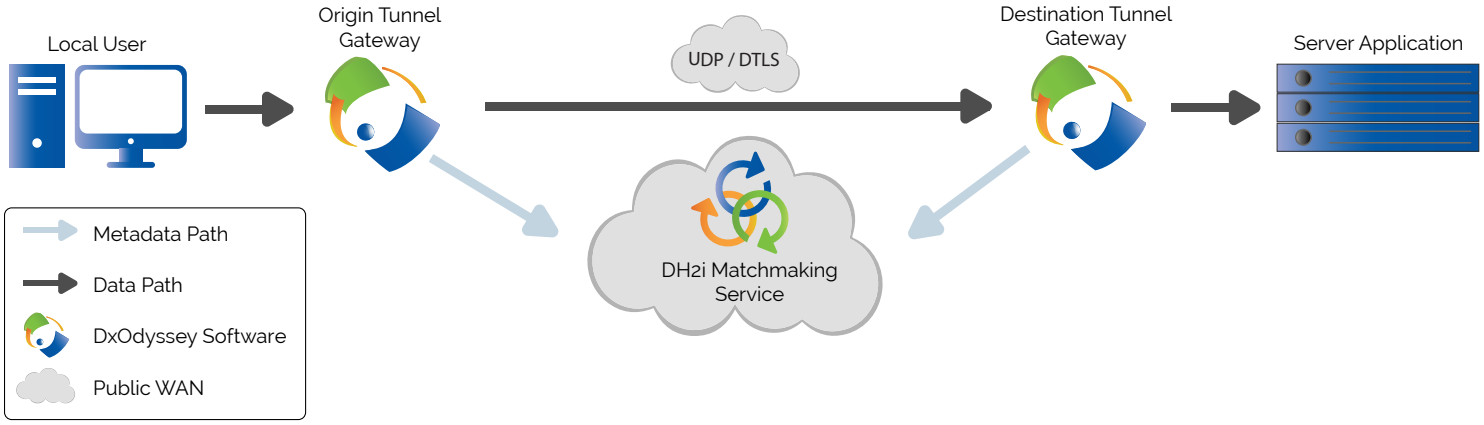
» The extensible nature of PostgreSQL on Ubuntu is ideal for enhancing its utility with a number of add-ons and extension hooks critical to decentralized applications. Extensions like the PostGIS geospatial database extender enable Geophysical Information System (GIS) mapping so effective that it seems native to PostgreSQL. With users in the field accessing this database for these distributed use cases and others, it becomes imperative to adequately protect data transmissions from remote locations with modern security methods designed for such extensible computing necessities.

Real-Time Processing

» Many of the sharding techniques for which PostgreSQL is reputed assist with the low latency required of transactional systems, real-time analytics use cases, and others. PostgreSQL nodes can deliver sub-second performance for analytics and dashboards, at scale, where the data are. Still, implementing these clusters in such highly distributed settings (like retail locations across the country or state, for example) decentralizes data transmissions while driving the need for cyber security at these decentralized locations. Real-time systems may also have users in the field accessing them, which adds to the endpoints organizations must protect.

The Solution

The compartmentalized micro-tunnels of DxOdyssey offer invisible data transmissions between virtually any setting without exposing data to anyone other than desired target systems. Organizations can follow these simple steps for securing DxOdyssey's segmented tunnels between distributed locations:



» *Architectural diagram of a secure micro-tunnel created with DxOdyssey*

How to Access PostgreSQL on an Ubuntu Node Using DxOdyssey

1. Install DxOdyssey on the nodes you want connected.
2. Link them with a matchmaker tool assigning random port generation for heightened security.
3. Put the nodes for the gateway and database in each other's host files.

If PostgreSQL is already installed on an Ubuntu node:

4. Ensure a user has permissions to access the database. This may require creating a specific user or relying on the default user 'postgres'.
5. Ensure the configurations file, (/etc/postgresql/9.5/main/postgresql.conf), has the listen address option set to "*".
6. Ensure the pg_hba configuration file has the line "host all all 0.0.0.0./0 md5". It may be necessary to comment out the "local all postgres md5" line if it appears.
7. Restart the service if you changed the configuration line.

If PostgreSQL is not previously installed:

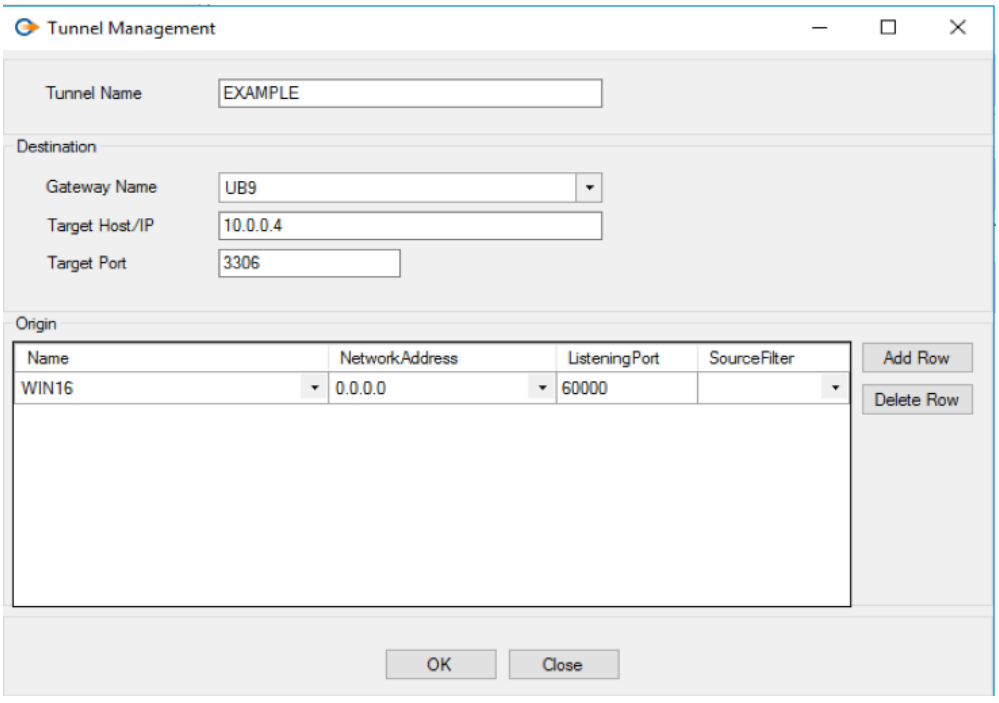
4. Install PostgreSQL via: sudo apt-get install postgresql.
5. Log in to the PostgreSQL shell and create a password for the user "postgres".
6. Go to step 4 of 'If PostgreSQL's already installed on an Ubuntu node' above.

8. Ensure the Ubuntu firewall allows connections to the appropriate port by first: finding the port (you can check the port in service configuration files) the service is on in the firewall settings. Make sure incoming connections are allowed. PostgreSQL default: 5432.
9. If it's not allowed, allow it: sudo iptables -A IN_public_allow -p tcp -m tcp --dport PORT_NUMBER -m conntrack --ctstate NEW -j ACCEPT.
10. Create the tunnels in DxOdyssey by first: clicking on Tunnel Manager.
11. Click on Add Tunnel.

STEP-BY-STEP INSTRUCTIONS CONTINUED ON PAGE 4

12. Name your tunnel.
13. Pick your gateway node from the dropdown menu.
14. Enter your database node's IP address in the Target Host/IP field.
15. Enter your database's port number in the Target Port field.
16. Add and configure your origin node(s). Remember the listening port you choose for your origin nodes.
17. Click Ok to add the tunnel.

The process generally takes less than five minutes if PostgreSQL is already installed in Ubuntu. By transporting data between gateways via UDP modified for packet control, DxO's invisible tunnels deliver application level security enhanced by DTLS encryption and public keys. Even other applications on the network won't be able to detect transmissions.



» *You should now be able to access your database through DxOdyssey tunnels by connecting to the origin node on the specified port number*

The Benefits

The benefits of using DxO tunnels to access PostgreSQL on an Ubuntu node are primed for the horizontal scaling and decentralized nature of this database. They include:

Software Defined Perimeters

» Much more flexible and easy to deploy than traditional cyber security perimeter methods are, software defined perimeters secure data between applications so that other users on the network—and intruders—aren't even aware decentralized applications are exchanging information. These cloaked application exchanges offer more granular security than traditional network security options do.

All-Inclusive Layered Security

» The architecture of DxO tunnels includes layers of security to protect data assets. Applications are connected between multi-tenant or remote users via a cloud matchmaker service that randomly generates a port for each gateway to access. Data is transported between the ports via packet-secured UDP, which is more difficult to detect than its TCP counterpart is. Ports are closed once the applications are connected, while data inside the tunnels are protected via DTLS encryption and Public Key Access mechanisms.

Decreased Database Costs, Management, and Administration Concerns

» Deploying DxO's invisible micro-tunnels increases the value of PostgreSQL nodes in remote locations. Organizations can leverage this same database with concurrent users without worrying about individual administration, management, and db expenses for each location from which it's accessed. This approach is effective for reducing database costs via sharding between remote locations, which simultaneously decreases response times for low latent applications while offering secure transmissions between them.

Massively Parallel Multi-Tenant Processing

» Organizations can distribute applications in multi-tenant settings and realize the performance gains of PostgreSQL via secure data transmissions without altering existing architecture. Users won't have to compromise security or expand their attack surfaces to avail themselves of distributed computing advantages.