NETFOUNDRY

STOP DATABASE BREACHES WITH ZERO TRUST ACCESS IN JDBC

Data is more distributed

Data is the life blood of every organization. Data was traditionally stored in centralized databases running in corporate controlled data centers. That reality has changed dramatically. Today, organizations are building and deploying composite applications that combine existing and new functionality, often cloud based, to support new user experiences and digital business opportunities. Composite applications access data stored in multiple databases, from multiple sources and providers, in multiple clouds, and increasingly from highly distributed edge environments.

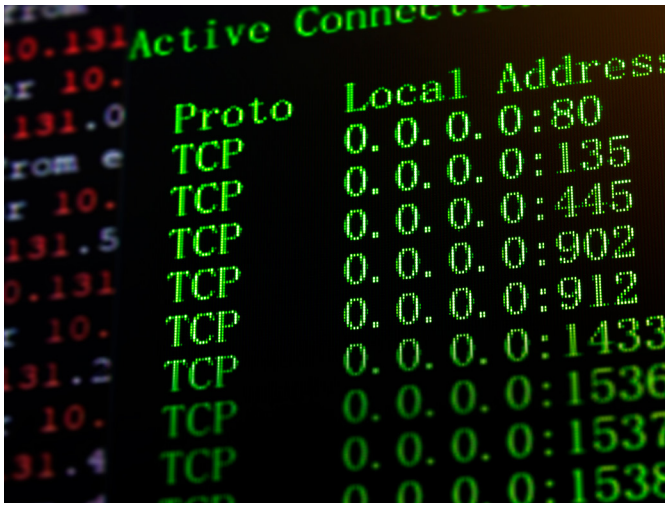
Securing composite applications and the data they use is extremely difficult and complex. Security professionals will tell you – complexity creates risk. A recent study found that almost half of internet facing databases (more than 12,000 in total) currently have vulnerabilities with a CVE rating of as "high" or "critical" while another study found 80% of internet exposed services are compromised by threat actors in under 24 hours. **Clearly, exposing your database to the internet is bad!**

Any inbound port is a risk to your business.

Access to databases is typically done using JDBC defining how a client may access the database. As businesses, wisely, do not want to expose their databases to the internet, they normally implement traditional methods such as Firewalls, VPNs, and dedicated private circuits (MPLS).

Unfortunately these "solutions" introduce additional complexity and vulnerabilities while slowing down innovation and creating a poor user experience:

- Open Firewall ports and public IPs are prime targets for DDoS and brute force attacks
- Maintaining firewall rules adds complexity and has a high potential for errors
- VPNs are cumbersome to set up and maintain, and deliver a poor user experience
- Users may seek "alternate" methods for accessing databases to get around constraints imposed by VPNs
- It is common to experience performance problems with VPNs - see for yourself



Contact us Share

For more information, visit us at www.netfoundry.io

NETFOUNDRY

This generic diagram is similar to how a cloud-native data warehouse platform recommends options to provide 'private' connectivity to their cloud hosted solution - it looks complicated because it is complicated! Adding complexity, increasing costs, and consuming valuable precious human resources to manage these systems is not a recipe for success. Some companies have partially solved the problem, e.g., Tableau, with their Bridge solution, but we need to be able to apply this to every scenario and use case while making both sides of the connection outbound.

What we really need is an open-source, 'clientless' solution which can be bundled into the applications leveraging JDBC to embed private, software-defined, zero trust connectivity. With this sort of solution, all databases and applications can be 'dark' to the internet (i.e., no public IP or inbound ports).

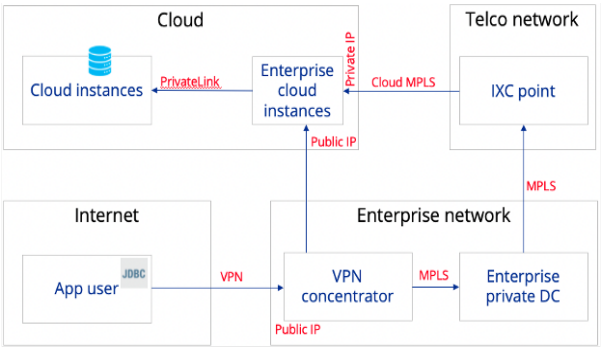


Diagram 1 - Suggested 'private' connectivity options

Step forward Ziti

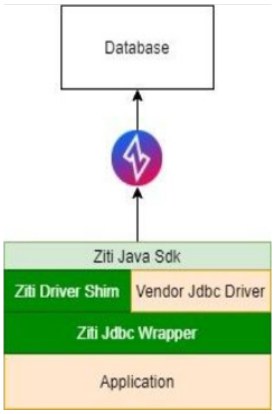


Diagram 2 - Zitified JDBC options

Luckily, the solution needed and described above does exist, provided by NetFoundry, built on OpenZiti.

By building Zitified JDBC (or ZDBC as we like to call it), we enable anyone to use **private, outbound-only connectivity** between their databases, applications and users **without modifying existing vendor drivers, application code, or the database** making it is simple to use with any customer application or third party tools.

As any communication to the database mandates usage of a database driver and API, ZDBC effectively gives us 'clientless' zero trust - just drop in ZDBC instead of JDBC/ODBC.

This plays out in two scenarios:

1. User to database access - for example, a data scientist using a data analysis tool to directly query some data source. That tool uses JDBC drivers, provisioned by the user, IT and/or the solution provider. Now they just use ZDBC.

2. Server to database access - for example, Oracle web logic is directly talking to backend databases. Instead of using JDBC, now it uses ZDBC.

Secure while subtracting

While practitioners will love the fact that they do not need to modify their code. Security will adore outbound only connectivity so that apps and databases are dark to the internet. What other activities no longer need to be done?

- No need to deploy and maintain VPNs, jump servers, bastion, MPLS, or static access controls as well as DDoS appliances, IPS/IDS, DNS trickery or SAML integration
- No need to open different inbound ports or maintain complex firewall rules - you only need outbound 80, 443, 6262
- If your vendor has embedded Ziti into their solution, you no longer need to maintain a cloud account and private connectivity (e.g., AWS PrivateLink or Azure Private Endpoint)
- No need to worry about what internal and external bad actors can access, the default is nothing. You set the policy and see access to resources based on identity (rather than IP) - check out the NetFoundry blog on how we use Ziti to protect our DevOps environment
- No need for users to maintain a connectivity client on their device (e.g., VPN)
- No need to build and maintain your own solution (e.g., Tableau Bridge)

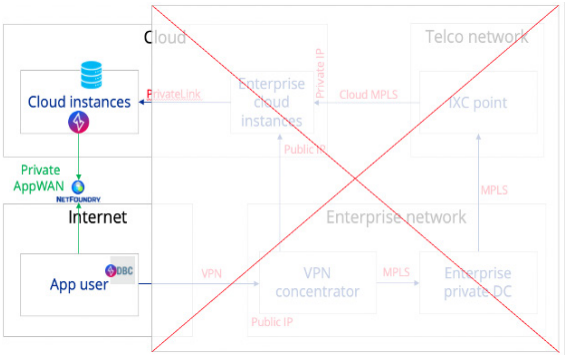


Diagram 3 - Cloud-native private connectivity

Contact us

Share

For more information, visit us at www.netfoundry.io

NETFOUNDRY

The 1,2,3 for Securing Database Access

Step 1: Seal off your database - deny all inbound ports on your firewall

Typical firewalls have thousands of rules, exceptions and policies. Replace them with one rule - **deny all inbound sessions**. Cyberattacks can no longer get to your database.

Step 2: Secure access for data clients – Apply Zero Trust to the JDBC/ ODBC API

Provide your users and applications with the Zitified JDBC driver. This wrapper incorporates zero trust capabilities into the driver and it can now access network resources securely from anywhere in the world, provided that they have internet access.

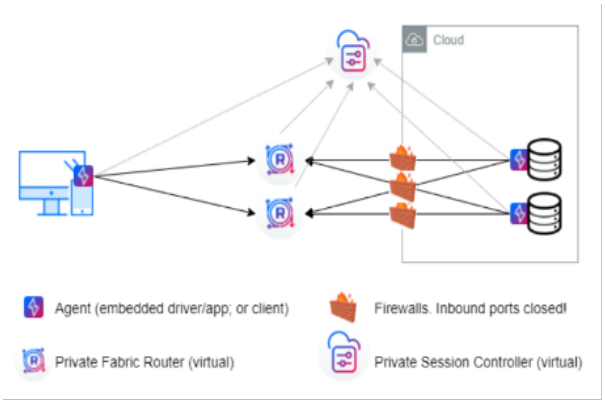


Diagram 4 - How Zitified JDBC architecture looks

Step 3: Private Fabric Brokers

With inbound ports closed, how do we access the database? ZDBC endpoints open outbound-only sessions to Private Edge Routers, hosted by NetFoundry, using our system of embedded identity (based on X.509). All other attempts to access the overlay network are denied. In fact, the entire network and your databases are invisible to outsiders. Malware cannot attack something it cannot see.

Go Open Source

We aim to cover the world in Ziti, that's why we open source it and its components - including the ZDBC wrappers, here is the Zero Trust wrapper for JDBC. We also provide open SDKs options to embed Zero Trust directly into your apps, and any device, so that you have full flexibility to choose whatever is simplest for you.

Check out and contribute to the OpenZiti project to help the world be more secure and simple by design!

ADDITIONAL RESOURCES

OpenZiti Project: https://openziti.github.io/index.html

Video: Secure Access to Postgres: https://www.youtube.com/watch?v=k2KlFXDQxvo

Web: https://netfoundry.io/

Blog: https://netfoundry.io/about/blog/

Twitter: @NetFoundry

LinkedIn: https://www.linkedin.com/company/netfoundry/

ABOUT NETFOUNDRY

NetFoundry is the leader in Cloud-Native Networking, enabling businesses to simply, securely and cost-effectively connect distributed applications across edges, clouds and service meshes. The NetFoundry platform, delivered as SaaS, enables businesses to connect applications without the costs and complexity of VPNs, custom hardware and private circuits.

NetFoundry's platform is accessed via APIs, SDKs and DevOps tools integrations, enabling practitioners, application developers, and network administrators to get the levels of automation and agility which are only possible with connectivity-as-code. NetFoundry is headquartered in Charlotte, North Carolina, with offices in California, Colorado, New York, London, Bangalore, and Singapore.

Contact us Share

For more information, visit us at www.netfoundry.io