NetFoundry™

SPIN UP YOUR NETWORK

Zero Trust API Security Cloud

Contents

Zero Trust API Cloud to make Public API Endpoints Private

APIs are exploding at 200% annual growth rates, and securing these APIs is very difficult. In fact, Gartner predict APIs will be 2022's most frequent cyberattack vector. Securing APIs is difficult because:

1. It is often infeasible to ask all your API clients to use private network solutions like VPNs or MPLS – the clients are too numerous, distributed and dynamic. The result is most API endpoints are accessible from anywhere on the Internet.
2. Newer tools such as zero trust security clouds (Zscaler etc.) are built to secure user-application access, rather than API networking.
3. API gateways and WAFs provide layer 7 security, but are still exposed to attacks from the network (layers 3 and 4) because VPN/MPLS can't be used. This enables attackers to target authentication and authorization bugs, zero-days and misconfigs…from the Internet. The result is 7 of the top 10 OWASP API vulnerabilities are not addressed, and they all can be attacked from the Internet.

To address this API security problem, NetFoundry provides customers with a Zero Trust API Cloud. The solution is similar to the zero trust clouds being rapidly adopted to secure user-app access (Zscaler etc.), but is built to help secure APIs against the top OWASP vulnerabilities (detailed in the next section). The Zero Trust API Cloud:

1. **Simplifies API security**. Replace the complex firewall ACLs with one inbound rule: deny-all. Fits in with existing WAN and API gateway architectures. No infrastructure deployment.
2. **"Shift left"**. The unique agentless approach enables developers to embed private, zero trust overlays in API client endpoints, as code, rather than needing separate VPN clients or firewall entries for each API client (code example).
3. **Comprehensive**. Includes mTLS (with enrollment, PKI infrastructure and key management, (and standards-based options to integrate 3rd party CAs and identity solutions); identities; optimized network routing and management functions.
4. **Provides the flexibility of SaaS and open source options**. The platform is available both as hosted SaaS, and as an open source platform for self-hosting. The cloud-native, API-first approach enables customers to leverage existing solutions and tooling, in both options.
5. **Helps mitigate solution sprawl**. The platform secures the APIs, and secures remote management of the API infrastructure, including both admin access and CI/CD system access. This centralizes identities and policies, and eliminates the need for bespoke solutions for different use cases.
6. **Improves visible**. The platform provides app-level network visibility to all API Producer internal stakeholder groups, and even to the end customers (API consumers), if desired. The rich data is exposed via both web console and NetFoundry Management APIs.

This paper discusses the architecture of the NetFoundry Zero Trust API Security Cloud and customer case studies.

Helping API providers better mitigate against the top 10 OWASP API threats

The NetFoundry Zero Trust API Cloud enables API providers to better mitigate against the problems identified by OWASP as the most severe API security vulnerabilities. In summary:

- 7 out of 10 of the OWASP top 10 API threats are **not** protected against by network firewalls (FWs), Web Application Firewalls (WAFs) or API Gateways (GWs). This is one reason why these threats are in the top 10!
- FWs, WAFs and API GWs **are** helpful with OWASP #4. WAFs and API GWs are effective for a **subset** of OWASP #7 and #8.
- NetFoundry can prevent **over 95%** of OWASP API threats #1, #2, #7, #8, #9, #10.
- NetFoundry **reduces the attack surface by factors of 8 to over 2200** for OWASP #3, #4, #5, #6 (the 8 – 2200 range of attack surface reduction depends on how many of the attacks are from the outside, as opposed to from authorized API clients, and this also varies by attack type).

Network firewall, WAF, API gateway	NetFoundry Zero Trust API Cloud
OWASP #1 Severity API Threat: Broken Object Level Authorization **OWASP #2 Severity API Threat: Broken User Authentication**	
• Network FWs only prevent attacks from known attackers, and only after their IPs are added to the ACL. • WAFs only work if the auth problem is previously known and widespread enough to be added to the WAF (usually not the case and after initial damage). • API GWs which provide mTLS user authentication and basic OpenAPI schema validations can be helpful, but are insufficient.	• NetFoundry proactively shields the endpoint which is exposing the attacked object – the vulnerability can't be exploited from the networks. • NF adds independent authentication and authorization, so the attacker needs to simultaneously break the API auth, NF's API auth, and gain access. • NF's mTLS is key in protecting the API provider (as opposed to TLS which primarily protects the API consumer).
OWASP #3 Severity API Threat: Excessive Data Exposure	
When the API developer needs to expose too many object properties, or rely on the API client to do all the filtering, then none of these solutions are very valuable for authorized users, beyond very elementary detection of sensitive data or schema deviances. However, NetFoundry uniquely ensures that only authorized API clients can enter, limiting the attack surface. If an authorized API client exploits this vulnerability, then NF 'quarantines' and sometimes disarms the threat entirely – it can't attack laterally, 'phone home', etc.	
OWASP #4 Severity API threat: Lack of Resources & Rate Limiting	
This can be addressed by all the solutions. NetFoundry simplifies this problem by limiting the attack surface – eliminating all the requests from unauthorized API clients which add noise for the API operators if they are only relying on the other solutions.	
OWASP #5 Severity API Threat: Broken Function Level Authorization **OWASP #6 Severity API Threat: Mass Assignment**	
If the API developer needs to implement complex access control policies, or use mass assignment to bind client data and data models without granular filtering, then it is mainly up to the	

<table>
<tr><td colspan="2">developer to prevent authorized users from finding holes or modifying object properties. Network firewalls do not help with these issues, and WAFs and API gateways can only do basic schema validation, when applicable (e.g. OpenAPI).

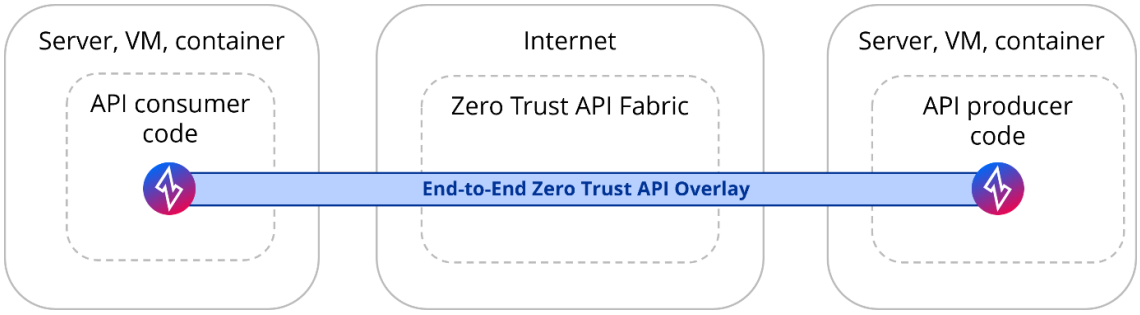
However, NetFoundry helps by minimizing the attack surface and noise – ensuring that unauthorized attackers can't take advantage of function level authorization issues.</td></tr>
<tr><th colspan="2">OWASP #7 Severity API Threat: Security Misconfiguration</th></tr>
<tr><td>Network FWs are irrelevant in dealing with these misconfigurations.
Some WAFs or API GWs may be able to detect commonly misconfigured HTTP headers, unnecessary HTTP methods, or permissive Cross-Origin resource sharing (CORS), but are insufficient.</td><td>NetFoundry proactively shields the misconfigured endpoint so the vulnerability can't be exploited from the networks, and the developer has time to fix it.
NF minimizes the blast radius. If a misconfiguration is exploited by an authorized API client, and it results in malware which then needs use the network to do damage, NF effectively quarantines it.</td></tr>
<tr><th colspan="2">OWASP #8 Severity API Threat: Injection</th></tr>
<tr><td>Network FWs are irrelevant in dealing with injection threats, other than very basic WAF rule sets.
Some WAFs or API GWs may be able to detect some common injection flaws, such as SQL, NoSQL, and Command Injection.</td><td>NetFoundry proactively shields the API endpoint or gateway from the network, such that the injection flaw can't be exploited from the networks, and the developer has time to fix it.
NF minimizes the blast radius. If an injection flaw is exploited by an authorized API client, and it results in malware which then needs to 'phone home' in order to do damage, NF effectively quarantines it.</td></tr>
<tr><th colspan="2">OWASP #9 Severity API threat: Improper Assets Management</th></tr>
<tr><td>Network firewalls, WAFs and API GWs are largely irrelevant in the context of asset management attack vectors (these debug endpoints etc. often look the same as the production API infrastructure), although they can help with API discovery.</td><td>NetFoundry also may not be able to differentiate between these endpoints and prod, but NF proactively shields all endpoints from network-based threats, therefore protecting against these threats.</td></tr>
<tr><th colspan="2">OWASP #10 Severity API Threat: Logging and Monitoring</th></tr>
<tr><td>Network FWs, WAFs and API GWs are largely unhelpful for the management threat because they are mainly outside of the interaction between API infra and management systems, although can integrate with some SIEMs. Furthermore, those systems can amplify the problem because they don't block unauthorized users from trying to access the API at L3/L4. The resultant noise created is one reason why API</td><td>NetFoundry helps in multiple ways:
API producers extend their zero trust network to include monitoring, management and logging systems. This means attackers can't 'move laterally' between these systems, preventing both infection and spread.
Because NF blocks unauthorized users before they reach the FW, WAF, API</td></tr>
</table>

breaches often take over 200 days to identify, and often are found by third parties rather than internal teams.	GW or API server, the amount of logs to analyze can be decreased by 8x to over 2200x.

How NetFoundry helps enable API producers to strengthen security

Private networks defined by 'boxes and wires' (network firewalls, MPLS circuits, VPNs, etc.) often can't handle the scale, distribution and dynamics of APIs. This forced API providers to make most APIs be Public Internet facing. This shoved API Gateways, WAFs, servers and endpoints on to a playing field which is severely tilted against them because they are force to face attacks from the entire Public Internet, and they need to win 100% of the battles.

NetFoundry tilted the playing field back in favor of the API providers, enabling API providers to take their API endpoints off the Internet – to leverage private API networks, while API clients could still use Internet access. Private networks defined by boxes and wires can't handle large scale API deployments, but clever software can! It looks like this:



Yes, some software (code) added to the existing API code results in the API provider being able to close all their inbound firewall ports, and instead open outbound, mTLS-secured connections towards a private API Fabric. Let's take a look:

1. API endpoints (consumer and producer endpoint) are enrolled with identities integrated into a PKI. This is similar to the process used to get L7 auth tokens which In this case, the identities will be used to govern layer 3 (network).
2. The OpenZiti SDKs enable developers to add a few lines of code to their APIs to authenticate and authorize each API session, leveraging the enrolled identities, and route authorized API sessions across the Fabric. AWS Lambda webhook example:

```
#Takes list of messages and sends them to a mattermost server
def sendToMM(messages):
    import openziti
    # url = "[redacted]"
    url = "[redacted]"
    headers = {'Content-Type': 'application/json; charset=UTF-8'}
    if(len(messages)>0):
        print("Length of messages: " + str(len(messages)))
        for message in messages:
            values = '{"text": "'+message+'"}'
            values = values.encode('utf-8')
            with openziti.monkeypatch():
                response = requests.post(url, headers=headers, data=values)
                print(response.status_code)
```

3. The Zero Trust API Fabric is a private, cloud native, software defined network. Businesses can choose to use the Fabric hosted by NetFoundry as SaaS, or use the OpenZiti open source to host their own Fabric. Only authorized sessions can use the Fabric, and both the API producer and consumer endpoints open outbound sessions to the Fabric, enabling them to close their inbound firewall ports.

Case studies

Edge to cloud APIs
Private 5G (Microsoft)
Edge compute (Arrow)
IoT analytics (TOOQ)

In the Microsoft and Arrow cases, a NetFoundry agent is used on the edge servers. TOOQ uses an agent on Nvidia Jetson and Raspberry Pi devices. Regardless of agent form factor, it enables the edge services to leverage APIs without opening any inbound firewall ports. The remote side is secured against MitM type attacks because it is very difficult for a MitM attack to steal or mimic the locally installed X.509 certificates, and the attack can't even get on to the private API network in the first place without that certificate. mTLS secures both the client and server side.

API server to API server: B2B, multicloud and hybrid cloud
MPLS replacement (FWD Insurance)
Private APIs (Oracle)
Multicloud (IBM)
Hybrid cloud (CERM)

These use cases are difficult due to multiple administrative domains – either multiple businesses in B2B, or different orgs in the same enterprise (querying a shared API). The key capability to overcome this problem is embedding the zero trust software into the app (example: a Java app) or API (example: an AWS Lambda webhook) so that one org doesn't need to make another org run an agent or gateway.

Management solutions and database access
Zero trust user and admin access (Ramco)
Private Kubernetes (Ozone)
Multicloud SAP remote management (Novis)
Zero trust MSSP services (Ohka)

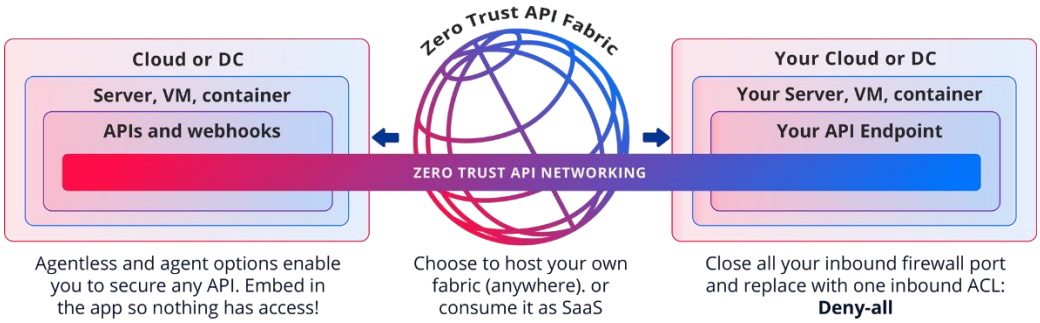
There are a few important trends across these use cases:

1. Distributed API queries are multiplying as app and API servers are moving closer to end users, while datastores remain more centralized. Suddenly, these queries are traversing Internet instead of racks in the same data center. Agents and VPNs are often not viable for security, performance and management reasons, making the app-embedded deployment very valuable.

2. As a site-to-site VPN replacement for securing APIs, NetFoundry enables the orgs to get stronger security (zero trust), without the complexity and operational overhead of VPNs for remote users, IoT devices, edge servers and third-parties.
3. The rash of recent damaging attacks of zero-day vulnerabilities on widely deployed self-hosted software demands a better solution. Even if the user to app segment is secured, the enterprise needs to secure APIs, remote management and server-to-server data in order to completely secure the app server (make it unreachable from the networks). For example, NetFoundry's customers running self-hosted Confluence servers were protected from the massive Confluence attack.
4. Management scenarios have always been sensitive due to the elevated levels of access which admins have. Attacks such as Kaseya have created further awareness to the point at which the MSP or MSSP is proactively deploying zero trust solutions between the provider's APIs and their customers. With environments far more distributed, and the addition of CI/CD systems (example: Ansible) needing secure remote management access, the VPN and bastion type solutions are often not secure or efficient enough.

Summary

Close all of your inbound firewall ports, making your API endpoints unreachable from the networks:



You can start now for free and have your first zero trust API up and running in less time than it takes to read this doc (if you don't skim read):

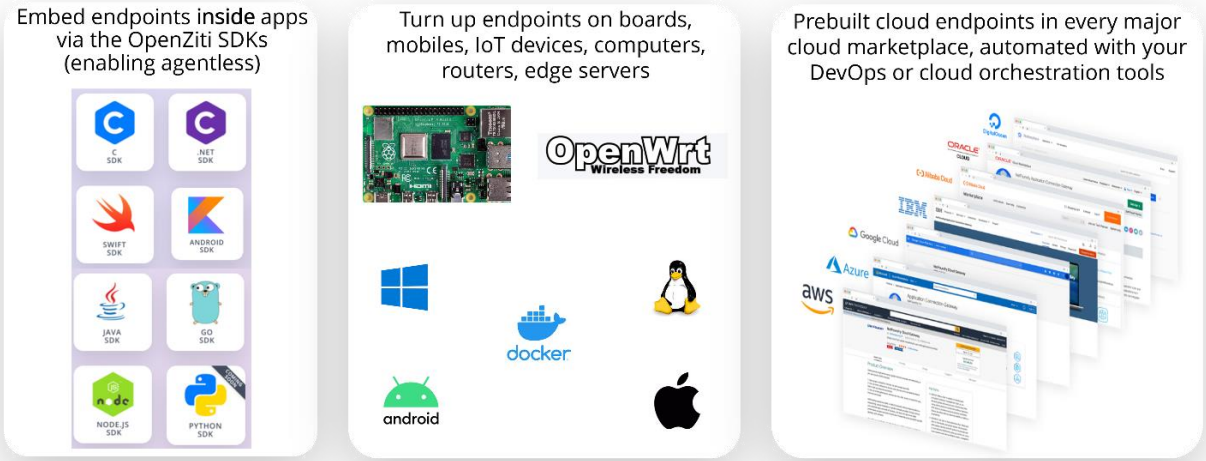
- NetFoundry SaaS (fully hosted), free forever for up to 10 endpoints.
- OpenZiti open source zero trust networking platform

Appendix 1: architecture details

Although this overview focuses on API and server-to-server use case, NetFoundry is a horizontal platform which combines zero trust functions with SD-WAN functions to enable enterprises to apply zero trust networking to any use case. This is critical in enabling a viable zero trust journey in which enterprises can prioritize specific use cases while not disrupting the rest of their topology, while unifying identities and policies, and providing coverage for any type of use cases, such that bespoke solutions are not required for future use cases.

Endpoints

There are three main types of endpoints:

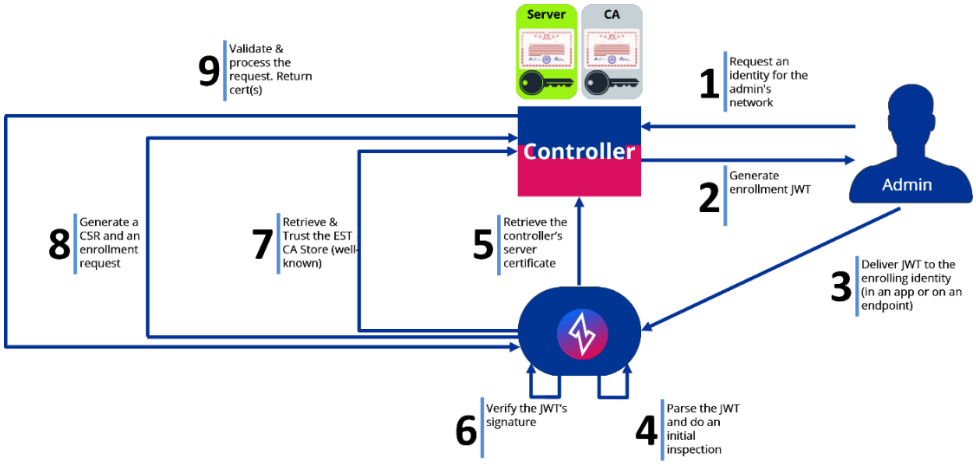


These endpoints go everywhere to make everything from remote management to APIs to databases be unreachable from the networks. These endpoints:

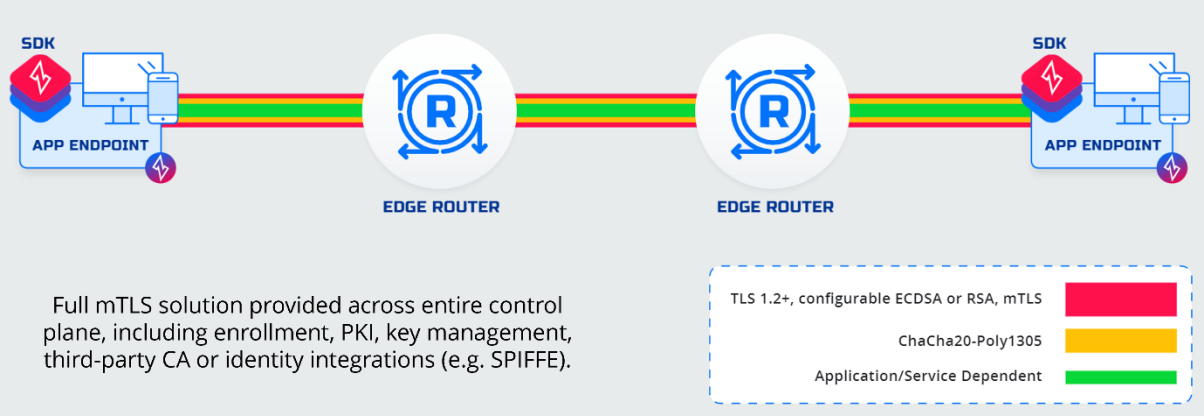
1. Contain X.509 identities and network stacks (to serve every protocol and use case), and give you a cloud-managed control point
2. Enable MFA and posture checks
3. Secure the app, data and device (make it unreachable from the networks).
4. Enable extensions into security and networking tools and solutions.
5. Provide networking capabilities such as routing, visibility and control.

Bootstrapped identification and authentication

Here is the basic diagram of the bootstrapped identity and authentication (further details are here):

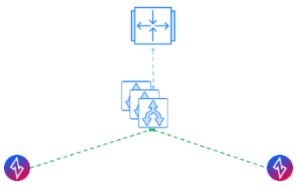


Endpoints need to be authenticated and authorized via their X.509 identities to access your private overlay. The bootstrapping and Certificate Authority (CA) are as SaaS, and you can add your own CA (via RFC 7030). The platform supports PKCS #11 - enables you to store certificates in secure ways (e.g. HSMs). Enjoy comprehensive mTLS security without deploying and managing PKI etc:



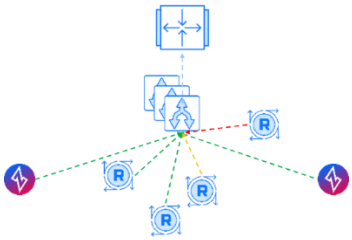
Least privileged access via distributed controllers

Your authorization and access policies are enforced via your private, NetFoundry hosted (in the SaaS model) or self-hosted (in the open source model) controllers:



Least privilege access can be paired with your IDP or SSO type solutions such that even if your IDP or SSO solution is compromised, there is still no network connection (layer 3). Similarly, if your NetFoundry solution is compromised, the attacker also often needs to thwart the IdP solution as well.

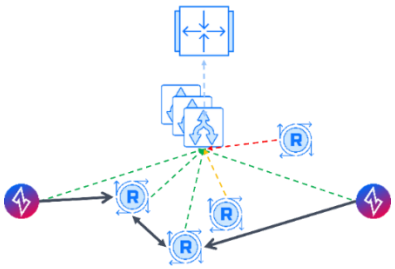
Private overlay fabrics



Your Routers function as combined routers/firewalls, but operate in the opposite model: instead of delivering packets unless told differently (IP based firewall rules), Fabric Routers deny all packets unless told differently (cryptography authorized flows). The Routers are hosted by NetFoundry in the SaaS, in every cloud marketplace and can be deployed as VMs anywhere. The Routers are governed by your policies (geofencing etc.) and are ephemeral – spin them up and down, programmatically. Your Routers form mesh overlay fabrics to:

1. Provide you with end-to-end control, across the overlay.
2. Enable you to close all your inbound ports. Instead, authorized NetFoundry endpoints will open outbound-only connections to authorized Routers. The Routers bridge the connections, enabling bi-directional data across the private overlay.
3. Provide you with optimized routing. This is detailed in the next section.
4. Enforce least privilege access on your overlay. This enables you to only grant permissions which are absolutely required for a given workload or session – no network level access. This applies to systems as well, for example a CI/CD system can only access certain ports on databases, and the connection will be ephemeral (only available during a code push).

Optimized, multipoint routing



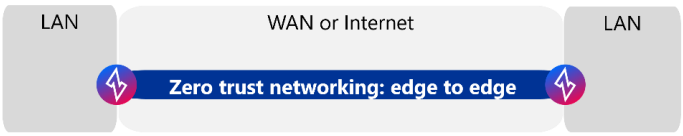
Real-time routing algorithms select the best path across your mesh, based on your metrics. In the example above, two of the four Routers are selected at first. This will change, automatically, as conditions change. Your Fabric Routers form your private, programmable Fabric, which can be used to:

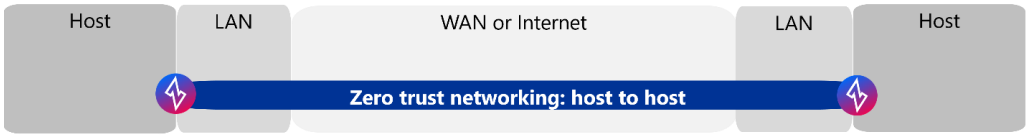
1. Enable your assets to deny all inbound connections, making your data unreachable from the networks
2. Enforce geofencing and similar policies. In the example above, the Router with the red connection was not eligible due to its location.

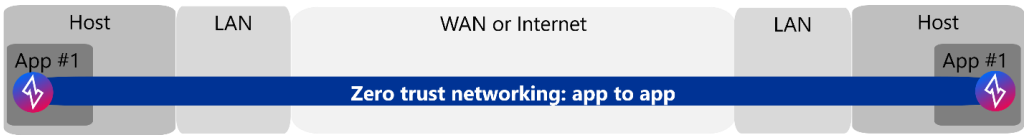
3. Use specific clouds for specific sessions.
4. Enable resiliency and improve quality (e.g. leverage routers across multiple edges and clouds). With Routers across multiple clouds and networks, there are many potential routes. When "Internet weather" makes certain routes perform better than others, or if certain routes have outages, the algorithms will automatically (per programmable policy) select the lowest latency routes.
5. Create ephemeral data planes (periodically spin up new sets of routers, creating a moving target).

Appendix 2: app-embedded, host-integrated, edge-integrated

A quick visual of each option: (from top to bottom): site to site; host to host; app to app:



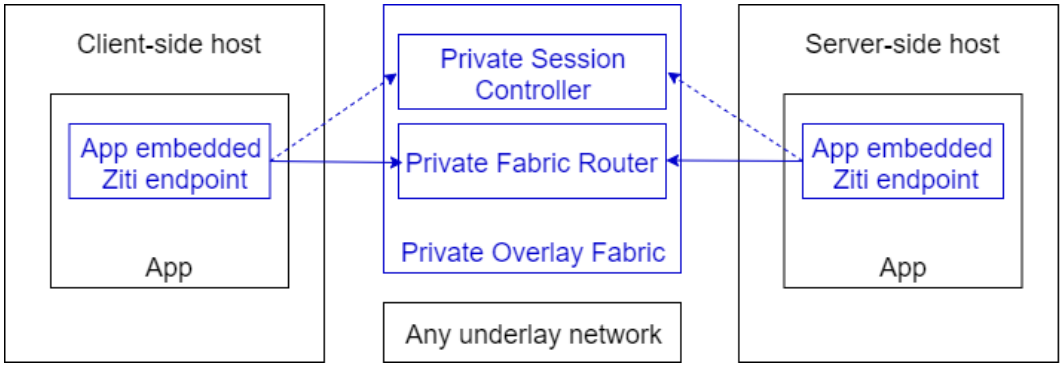




App embedded option (app, API, browser, proxy, agent, driver embedded)

Starting from the bottom of the diagram, app-embedded (agentless) solves these problems:

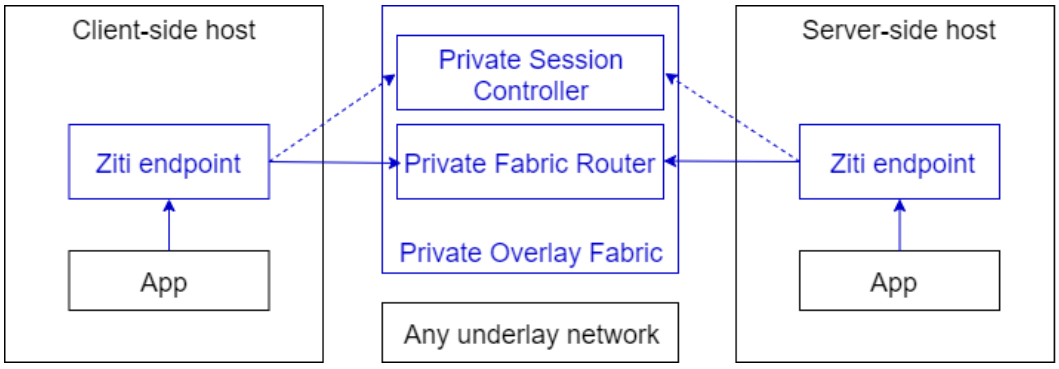
1. Security and control. Use cases which require the strongest security and control can now securely connect without even trusting the hosts.
2. Topology. Use cases in which it is difficult to deploy agents, including B2B APIs, third party endpoints, IoT and unikernal environemnts. By embedding in the app, data or database, secure networking goes anywhere your app goes, eliminating DNS, VPN, etc. problems (this video shows how to use the SDKs).
3. Integrations. Add code to existing security solutions, agents, proxies, browsers, API servers etc.



You can embed the software directly into the app, browser, proxy, API, database driver, etc (see the video in this blog post about "Zitifying" a Postgres driver).

Host integrated option (app specific agents for IT and IoT devices)

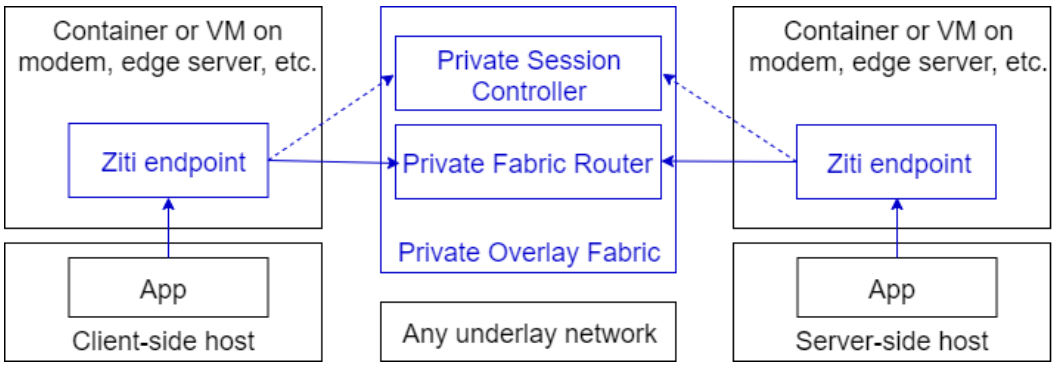
This option leverages NetFoundry software endpoints, which support every device type, OS and cloud, and are built on the OpenZiti SDKs described above:



Your endpoint is 'moved' from the app or browser to the device hosting the app.

Edge-integrated option (containers or VMs):

In this option, NetFoundry software endpoints are deployed as containers or VMs on modems, DMZ routers, edge servers and cloud instances:



This places your endpoints on aggregation points.