NetFoundry for secure RAG connectivity and networking

A key reason why NetFoundry securely delivers over one billion sessions per month is because NetFoundry helps **both** simplify operations and strengthen security.

This is important for distributed, dynamic applications which rely on sensitive data - which of course describes AI. This is why NetFoundry is used by enterprise AI teams for all AI phases, including fine-tuning, RAG and inference. This paper focuses on using NetFoundry to simplify and secure RAG. The key results are:

1. **Stronger security**. A secure by default environment in which LLM, data, and pipelines are "dark" to every network, and are instead microsegmented by identity - accessible only by the specific, authorized identities.
2. **Simpler operations**. No dependencies on IPs, clouds, networks, NAT, infrastructure or public DNS.

Secure by design RAG - example

This 5-page guide will take you through an example deployment. These are done in minutes once you are familiar with the software and terminology. For this example:

- Your LLM is in a private data center, such as Equinix, CoreWeave, Lambda, Vultr, RunPod or Paperspace by Digital Ocean.
- Your AI application servers and RAG orchestrator are in AWS.
- Your data, tools and CI/CD are in AWS and Azure.
- Your users and admins are everywhere.

The appendices have deeper compliance and technical details.

This is **not** a solution guide - it is not comprehensive and there is some pseudo-code. The purpose is to illustrate the deployment. Contact us if you want an architect to build a working lab environment for you, and then hand you the keys (for a free trial of up to 30 days).

Component #1: LLM in private data center

For this example, we assume your LLM is in a private DC, but the basic steps are the same if you deploy elsewhere, such as AWS, Azure, OCI or GCP.

Whether your LLM is on a bare-metal server or VM, you will close **all** the inbound ports on the server (e.g. host firewall, iptables or using NetFoundry's eBPF firewall). This makes your LLM unreachable from unauthorized endpoints - even those which are inside your private network.

Instead, your LLM will open outbound-only sessions, towards a NetFoundry tunneler. The tunneler is NetFoundry software which functions as a thin network endpoint, deployed either on your server or deployed as a gateway in your DMZ. The tunneler only permits sessions which match your policies.

The NetFoundry tunneler includes your choice of identity - X.509-based for proof of possession identity, and/or IdP based (see appendix for details).

Your policies define what your LLM can reach, and what services your LLM hosts. Your policies are all governed by your identities - not IP addresses - which is one reason why management is so simplified. We show these policies in detail in the component #5 section. As an example:

- **Service Name:** private-llm.api.service
- **Intercepted Address:** llm.local (this can be a "made-up" DNS name as NetFoundry uses private DNS)
- **Port:** 5001 (whatever port your LLM API serves on)
- **Host:** Bind this service to the new equinix-llm-endpoint identity
- **Host Address:** localhost or 127.0.0.1

Component #2: data, pipelines and RAG in AWS

AWS hosted data and pipelines

- **Components:**
  - Vector database (e.g., Amazon OpenSearch, Pinecone)
  - Knowledge graph (e.g., Amazon Neptune)
  - ETL/ELT tools (e.g., AWS Glue or services on EC2)
  - Data storage (S3)
- **NetFoundry software:** Add a NetFoundry Edge Router (ER) from the AWS Marketplace into your VPC.
- **Setup:**
  1. Configure all your AWS services (e.g. OpenSearch) to use VPC Private Endpoints so they have a private IP in your VPC and are not accessible from the Internet.
  2. Deploy the NetFoundry ER into the same VPC. The ER acts as the secure, zero-trust gateway for all services in that VPC.
  3. Define services (policies) in your NetFoundry console, "hosted" or “bound” by the ER identity. Example for one service:
     - **Service name:** aws.vectordb.service
     - **Host address:** private IP of your OpenSearch VPC endpoint.
     - **Port:** 443
  4. Repeat this for your other services. For example:
     - aws.kg.service (private IP of Neptune)
     - aws.etl.service (private IP of your ETL service)
     - aws.s3-data.service (private IP of your S3 endpoint)

AWS hosted RAG

In this example, the orchestrator service which runs your RAG logic (receives user query, queries data, queries LLM) runs on AWS on an EKS cluster or EC2 instance.

- **Component:** RAG Application Server (EKS pod or EC2 instance).

- **NetFoundry software:** Install a NetFoundry tunneler on the host VM or as a sidecar in the EKS pod.
- **Setup:**
  1. This endpoint registers with a strong identity, e.g., rag-app.identity. This identity is an X.509 certificate, with NetFoundry providing the PKI services (detailed in appendix).
  2. The RAG application code is written to make calls to the NetFoundry intercepted, private DNS names, e.g.:
     - http://llm.local:5001
     - https://aws.vectordb.service
     - https://azure.kg.service
  3. The local NetFoundry endpoint (tunneler) on the RAG app server will intercept these requests, encrypt them, and send them to the destination via your private, dedicated overlay fabric (hosted by you or NetFoundry).

Component #3: Data and pipelines in Azure

The result of this is the Azure data will not be reachable from unauthorized network endpoints. Likewise, even though the model can use the data, none of these components can initiate sessions into the model, meaning that the model is protected from a compromise of any of these components.

- **Components:**
  - Vector database (e.g., Azure AI Search)
  - Knowledge graph (e.g., Azure Cosmos DB)
  - ETL/ELT tools (e.g., Azure Data Factory)
  - Data Storage (e.g. blob storage)
- **NetFoundry software:** Add a NetFoundry Edge Router (ER) from the Azure Marketplace into your VNet.
- **Action:**
  1. Configure all your Azure services to use Private Endpoints, giving them a private IP in your VNet.
  2. Deploy the NetFoundry ER into the same VNet.
  3. Define **Services** in your NetFoundry console, hosted by the ER identity. For example:
     - **Service name:** azure.vectordb.service
     - **Host address:** The private IP of your Azure AI Search private endpoint.
     - **Port:** 443
  4. Repeat for other services:
     - azure.kg.service
     - azure.etl-runtime.service

Component #4: Users, developers and admins of your AI application and systems

This provides identity, authentication, authorization and reporting on all access, and securely connects all sessions, without any Internet exposure. This can extend to AI agents and MCP

servers as well, but that is a topic for another day (see this blog if you want a quick peek).

- **User types:**
  - Internal
  - External
  - Admins
- **Methods:**
  - **Clientless** - can be used for internal or external, but most often used for external access or cases in which thin clients on user machines are not practical. These sessions are managed via IdP-based authentication, provided as part of NetFoundry's Frontdoor service.
  - **Client** - usually mandated for privileged access and can be used for internal and external users as well (not intrusive for external - meaning it does not VPN tunnel all their traffic - it only impacts their sessions to your AI application). NetFoundry tunnelers for every major OS in each app store enable this access.
- **Purpose:** To allow users to use the app and developers to manage the infrastructure.
- **Identities.** Like all endpoints, the users and admins are governed by strong identities. These enable individual based or group based policies and can be integrated with your IdP or standalone. For example:
  - mlops-dev.group
  - end-user.group
- **Policies. Policies can be applied at any level of granularity but for example:**
  1. **End-User:**
     - end-user.group can **Dial** rag-app.https.service (the public-facing service for your RAG app, also hosted by rag-app.identity).
  2. **Developer or admin:**
     - mlops-dev.group can **Dial** rag-app.https.service.
     - mlops-dev.group can **Dial** aws-llm.ssh.service (SSH access to the LLM box).
     - mlops-dev.group can **Dial** aws-rag-app.ssh.service (SSH to the RAG app server).

Component #5: policies

You add your policies via NetFoundry's web UI or APIs. The result is separate, isolated, micro-segmented networks for each logical flow, defined at whatever granularity you want. NetFoundry can also help discover, recommend and vet policies.

Flow 1: RAG Inference

- **Purpose:** Allows the RAG application to fetch context and query the LLM.
- **Name:** rag-inference-flow
- **Identities:**
  - rag-app.identity (the RAG application itself)
- **Services:**
  - private-llm.api.service (Hosted in AWS)

    - aws.vectordb.service (Hosted in AWS)
    - azure.kg.service (Hosted in Azure)
- **Policy:**
    - rag-app.identity can **Dial** (access) private-llm.api.service.
    - rag-app.identity can **Dial** aws.vectordb.service.
    - rag-app.identity can **Dial** azure.kg.service.
- **Result:** The RAG app can perform its job. The LLM **cannot** initiate a connection back to the databases in AWS/Azure. The Azure Knowledge Graph **cannot** talk to the AWS Vector DB. You have one-way, least-privilege access.

Flow 2: data pipeline (e.g. ETL/ELT)

- **Purpose:** Allows your data pipelines to read raw data and write to the processed databases.
- **Name:** data-pipeline-flow
- **Identities:**
    - aws.etl.identity (Your AWS Glue job or EC2-based tool)
    - azure.etl.identity (Your Azure Data Factory runtime)
- **Services:**
    - aws.s3-data.service (Raw data)
    - aws.vectordb.service (Processed data destination)
    - azure.kg.service (Processed data destination)
- **Policy:**
    - aws.etl.identity can **Dial** aws.s3-data.service and aws.vectordb.service.
    - azure.etl.identity can **Dial** azure.kg.service.
- **Result:** Your ETL processes are isolated. They can *only* talk to the data sources and destinations they need. They don't have access to the private-llm.api.service. This prevents a vulnerability in a data pipeline from being used to attack or exfiltrate your private model.

Flow 3: User, developer and admin access

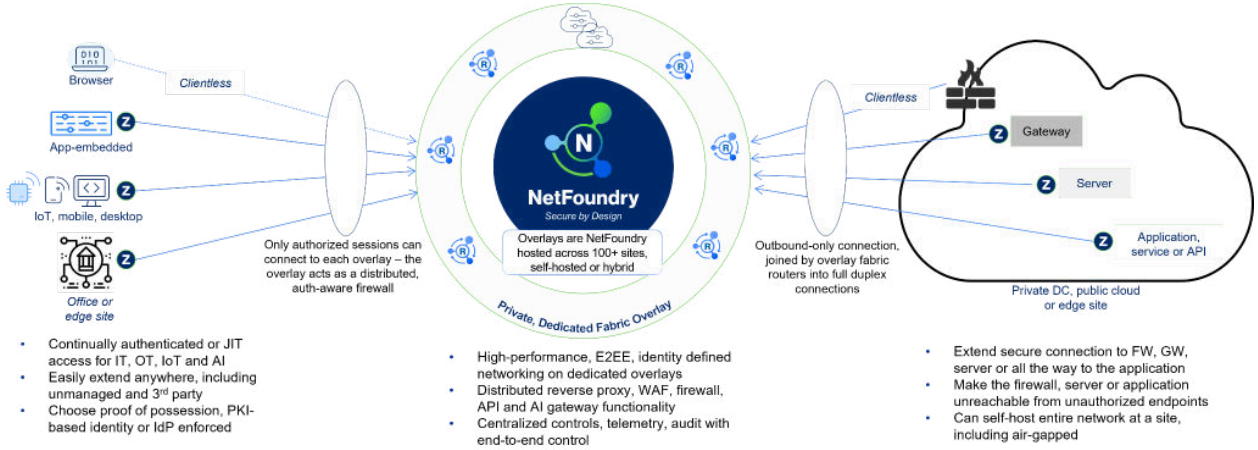
- **Purpose:** To allow users to use the app and admins to manage the infrastructure.
- **Identities:** see above in which we grouped individual identities into:
    - end-user.group
    - mlops-dev.group
- **Policies:**
    3. **User:**
        - end-user.group can **Dial** rag-app.https.service (the public-facing service for your RAG app, also hosted by rag-app.identity).
    4. **Developer or admin:**
        - mlops-dev.group can **Dial** rag-app.https.service
        - mlops-dev.group can **Dial** aws-llm.ssh.service
        - mlops-dev.group can **Dial** aws-rag-app.ssh.service
- **Result:** You have fine-grained, auditable, role-based access. Users can't even see the SSH services. Developers get ephemeral, just-in-time access to infrastructure without

needing a bastion host or VPN.

Appendix 1: compliance

Framework	Key Requirements Addressed	Technical Controls
HIPAA 164.312	Access control, audit controls, transmission security	IdP + MFA, NetFoundry mTLS, Elastic SIEM
GDPR Art 32/35	Data integrity & confidentiality, DPIA	Data encryption, overlay segmentation, logging
DORA Art 5/11	ICT risk management, third-party oversight	NF policy controls, vendor risk register
ISO 27001	Annex A – Information Security Controls	Vault secret mgmt, change mgmt, audit logging
FIPS 140-3	Cryptographic module validation	HSM and KMS use FIPS-validated modules

Appendix 2: NetFoundry's universal, secure-by-default architecture



NetFoundry enables businesses to use "secure by default" overlays in which all devices and services (internal and external) are strongly identified, authenticated and authorized before any communication, and no network ports are left open for reconnaissance, attack, lateral

movement or data exfiltration.

Identity-based authentication & authorization - overview

Two basic options for each session:

1. **Clientless**. The NetFoundry overlay uses an IdP to authenticate the user before the user is granted access. This is usually for 3rd parties and unmanaged devices.
2. **Client**. This includes proof of possession (PoP) identity, continuous authentication, granular authorization, MFA and posture. It is usually for managed devices and privileged access management. The 'client' can be on a device, at the edge of a network (gateway model) or built into an application (using NetFoundry's SDKs).

In both options, the server-side of the connection is no longer reachable from unauthorized endpoints. For example, the host firewall (iptables, eBPF, etc.) on the server has one rule: deny-all inbound. The attack surface is therefore changed from the Internet to insiders.

Identity-based authentication & authorization - clients

- Host-based clients are available for every major OS and cloud and are in all the major marketplaces. This includes OT, IoT, mobile, desktop and server
- Gateway versions are available as virtualized or containerized. They scale horizontally, each doing about 1 Gbps on mid-range hardware (e.g. medium sized cloud instances).
- Businesses provision clients via methods like SCIM and JWT (or build the clients into their software with NetFoundry SDKs).

Notes:

- NetFoundry clients enforce overlay policies – for example, posture and MFA are added for applications which don't support mechanisms like SAML, OAuth or OIDC.
- NetFoundry clients only intercept the applications which are defined in the policy. For example, a NetFoundry client on a third-party user device can deliver a specific API, while the rest of the sessions are unimpacted. This makes the clients usable by third parties, since they are not tunneling all traffic.
- NetFoundry clients route each session independently to its destination as 'direct flights' (using the best performing path), rather than VPN 'two stop flights' which tunnel all traffic to a central location, and then route it to the actual destination.

Identity-based authentication & authorization – client-based solution details

1. Every endpoint (user, device, or service) is enrolled via a unique, RSA or EC cryptographic identity (X.509 certificates), used for mTLS connectivity.
    a. Hardware root of trust or TSM can be used for key storage (PKCS11). This functions as a proof of possession (PoP) identity such that credential theft is thwarted – the attacker must take control of the machine to get any access – not simply use stolen credentials, tokens or assertions from the Internet.
    b. An IdP can be used instead or in addition to the X.509 certificate. The CA/PKI

solution is included in NetFoundry NaaS, and other CAs are optionally integrated via RFC 7030 support.

2. Connections are only allowed **after** authentication
    a. It is a continuous authentication model – meaning if there is a change in posture or policy during a session, then that session is terminated.
    b. 3rd-party CA, 3rd-party IdP, external JWT, username, password and TOTP are supported (RFC 6238 is supported), as primary or secondary authentication.
    c. Both side of the connection means both the client side and server side need to be authenticated.
3. All authorization is controlled by identities and policies – the default is no access
    a. This includes one-time access, just-in-time (JIT) access and continually authenticated access models. For example, a ticket system or CI/CD workflow can instantly create an identity or policy and instantly revoke it upon a certain event or timeframe.
    b. It is continuous authorization. For example, a policy change will terminate even an existing session.
    c. The client side needs to be authorized to connect to the service hosted by the server side identity, and the server side identity needs to be authorized to allow the specific client identity to connect.

End-to-end encryption & data cloaking

1. All data is secured with strong encryption (configurable ECDSA or RSA for mTLS 1.2+, ChaChar20-Poly 1305, libsodium). Other ciphers such as FIPS-compliant encryption and post-quantum encryption (PQE) are optionally plugged in.
2. Application data is end-to-end encrypted between the communicating parties – overlay routers or network admins cannot access, read or tamper with the data. This protects data sovereignty, even in the case of an attack or insider threat.
3. NetFoundry's encapsulation and encryption helps obfuscate metadata – packets look like TLS traffic on port 443 (or whatever port has been configured in NetFoundry), with no useful info in either content or headers for an adversary to exploit. This helps thwart reconnaissance (port scanning, protocol fingerprinting, traffic analysis, intelligence gathering)

'Dark' services, devices or networks

NetFoundry makes services, devices or sites unreachable from unauthorized endpoints. This shrinks the attack surface to insider attacks only. Attackers cannot access or even enumerate a NetFoundry-protected resource without being enrolled in the system and authorized for that specific session. NetFoundry secured resources will not respond to any unauthorized ping or connection attempt – the service, device or server might as well not exist, depending on which implementation:

- When using NetFoundry SDKs, an application, session or service is unreachable from anywhere – even hosts don't have access. For example, an AI agent can be scoped to not have any network access, and not to be accessible from the networks, but to only

have access to connect to specifically defined microservices.

- When using NetFoundry host-based endpoints, the endpoint (e.g. a server) can be made to only open sessions outbound to its private NetFoundry overlay and not listen for inbound traffic from the networks. A common use case is to make a server unreachable from the Internet or underlay network.
- When using NetFoundry's edge, site or cloud endpoints, the site devices can route outbound sessions to the NetFoundry site endpoint (it functions as the default gateway and network firewall), and to not accept inbound requests from the Internet or underlay network. A common use case is to make an edge or cloud site unreachable from the Internet (instead of relying on an Internet-reachable firewall). This supports remote access as the outbound session from the site devices supports full-duplex connectivity from authorized endpoints.

Identity based microsegmentation and least privilege

Through NetFoundry, administrators define fine-grained segments and access policies, with full visibility from telemetry and audit perspectives (NetFoundry helps discover the network flows and define policies for them).

Each application or device is segmented so they can't communicate, even if on the same physical network, unless policy specifically allows. The matching of authorized identities to services helps mitigate against malicious lateral movement and data exfiltration. This also sandboxes what a compromised node can do.

Adaptive, HA networking with end-to-end control and visibility

NetFoundry NaaS uses full mesh adaptive routing across over 100 data centers on the world's top IP backbones to optimize reliability and performance, with options to limit to certain geographies or providers. Each network is private and dedicated to its administrator, who can then decide to add tenants or clients.

Overlays can also be self-hosted, including in air-gapped sites. This overlay architecture helps NetFoundry provide end-to-end control and telemetry, especially because it can be used for any use case, including supply chain partners.

Overlay network controllers instruct the endpoints and routers on the best performing route for each session. Endpoints simultaneously use different paths for different sessions (no single tunnel backhaul), changing paths as necessary, and doing load balancing. Routers do not have access to the encrypted payloads. Every link is mutual TLS (mTLS).

NetFoundry endpoints leverage diverse transport networks – if a radio link is slow or down, traffic might reroute over a satellite or wired link. The architecture is designed for high availability; there's no single point of failure, and the system can tolerate outages or route around network damage. This provides resilience in DDIL conditions.