VAST

White Paper

VAST DataEnclave Architecture and Trust Model

Attestation, deployment, and audit for confidential AI workloads

Contents

The Two-Sided Trust Problem 3

What the VAST DataEnclave Is 4

Worked Example: Closed Models on Customer Infrastructure 6

- Step 1: Proving What is Running Before Anything is Released 6
- Step 2: Delivering the Model Without Exposing It 7
- Step 3: Decrypting the Model Weights 7
- Step 4: Using the Model with Confidential Business Data 7
- Step 5: Using Confidential Storage for Model Runtime State 8

Attestation, Verification, and Failure 9

- Attestation Sequence 9
- Attestation Failure or Release Denial 11
- Trust Assumptions and Limitations 11

Deployment and Use 12

- Deploying DataEnclave 12
- Deploying Workloads with DataEnclave 13
- Ongoing Management 14

Audit and Evidence 15

Removing the Trust Problem 15

The Two-Sided Trust Problem

Many of the most impactful applications of AI require bringing together two datasets owned by different parties: confidential business information and closed-weight frontier models. Combining the two requires a compromise: one of those owners must send its data to infrastructure controlled by the other, implicitly trusting that they will not mishandle, steal, or expose it. Given the fact that these datasets are often the core business of the companies that own them, that trust is rarely available.

To date, the solution to this trust problem is compromising in another dimension: reach. When an enterprise cannot send its data to a hosted inference API, and a model provider is unwilling to deploy its model in an untrusted environment, the two simply never meet. At best, the enterprise can run a less-capable open model within its own infrastructure and accept a less capable result, while the model provider limits itself to the workloads with lower-value, non-sensitive data. Both protect themselves by not trusting the other, and the resulting AI applications with the most to gain from frontier models simply cannot be built.

Encryption and multitenancy, while critical, do not resolve this issue of trust alone. Encryption protects data while it is stored and as it moves across networks, but AI workloads themselves—training and inferencing—only operate on unencrypted inputs and outputs.[1] While training and inferencing workloads are running, both parties' datasets lie unencrypted in memory of the GPU servers, allowing the operator of that infrastructure to inspect their contents. This gap falls back to trusting that the infrastructure operator will not mishandle this data.

The solution to this trust problem is to remove this implicit trust from the equation altogether. Instead of trusting the infrastructure operator, each dataset owner needs a way to verify that the environment in which they will run is secure. If the environment cannot prove that it will protect workload data at rest, in motion, and being processed, it must never receive sensitive data or decryption keys. And that proof must be verifiable by an explicitly trusted third party so that the infrastructure operator cannot forge it.

Confidential computing is a family of technologies that have long closed this gap for CPU-based workloads using encrypted memory and hardware that can prove what is running. However, until recently, AI workloads could not be run confidentially because the GPUs that enable them lacked the hardware support to attest and isolate their workloads. A CPU can hold confidential model weights in encrypted memory, securing them from external inspection. But those weights must be decrypted into GPU memory before they can process a query, and GPU memory historically offered no comparable encryption or protections from host administrators.

This changed as NVIDIA GPUs have adopted high-performance confidential computing hardware capabilities. Since the Hopper generation, GPUs have been able to protect their memory from the host, allowing CPU-level trusted execution to extend to include GPU memory and the buses between them. An enclave can now span CPUs and GPUs with both being able to securely attest to their configuration, making it possible to run an entire AI workflow within a single, verified enclave on otherwise untrusted infrastructure.

This document describes how the VAST DataEnclave integrates these confidential computing capabilities into the DataEngine runtime to solve this trust problem by eliminating the need for implicit trust. It walks through how DataEnclave ensures that even infrastructure operators cannot access protected data as AI workloads execute, and it describes how this protection is independently verifiable before any sensitive data is released. By walking through different AI workload scenarios, it demonstrates what this attestation proves, what happens when it fails, and the boundary between DataEnclave's guarantees and applications' own controls.

[1]Homomorphic encryption is an area of research that does allow models to operate on encrypted data, but it is not yet economically viable. Inferencing directly on encrypted data requires around 28x more GPU memory and runs between 70x and 130x slower according to a 2026 study by Cornell, Penn State, and NVIDIA researchers.

What the VAST DataEnclave Is

DataEnclave allows for deployment of AI models while ensuring they cannot be copied or accessed outside of authorized use cases. This is achieved through confidential computing that protects the data of two parties:

1. **Model providers** can securely deploy their proprietary models into an untrusted customer environment
2. **Enterprise organizations** can use proprietary models with their confidential enterprise data by never having their data leave their premises

This allows data and application owners to define the minimum baseline security posture of the environments in which they will run; if an enclave cannot meet those requirements, DataEnclave will not pull container images, retrieve any secrets, or run workloads. DataEnclave ensures that all data crossing untrusted infrastructure is encrypted to ensure it is never exposed, and sensitive data such as keys are only ever delivered into trusted enclaves that are isolated from the untrusted host.

DataEnclave establishes trust in these enclaves by following the IETF's architecture for remote attestation, RFC 9334:

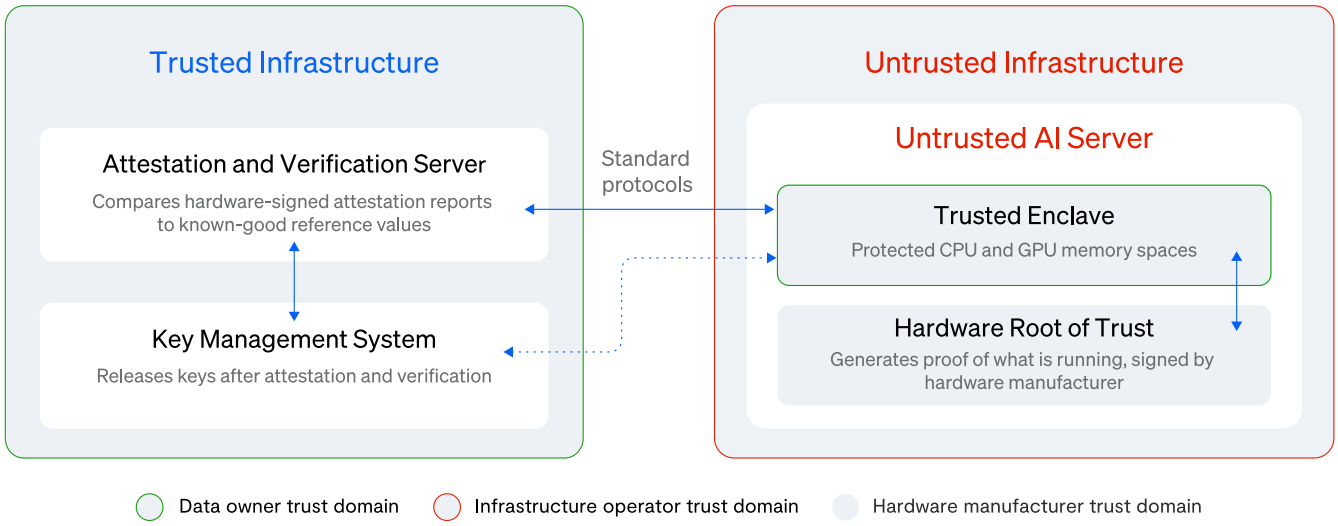


Figure 1 High-level overview of the DataEnclave confidential AI architecture.

DataEnclave provides two components that work with the data owner's key management system to protect confidential workloads and their data from the infrastructure they run on, including the host and its administrators:

- **The DataEnclave Runtime,** which runs within untrusted compute infrastructure and creates enclaves there. Individual enclaves are created as confidential virtual machines, DataEnclave CVMs, whose CPU memory is encrypted and protected from unauthorized access, and whose GPU memory and CPU-GPU communication are protected to prevent anything outside of the enclave from being able to see its contents. DataEnclave CVMs also contain agents to perform attestation and enforce runtime policies of their confidential containers. The DataEnclave Runtime is a required component and a core feature of DataEngine.
- **The DataEnclave attestation server,** which runs in a separate, trusted environment (e.g. model provider premises) and enforces the policies that govern which containers and secrets can be mapped to which DataEnclave CVMs. The DataEnclave attestation server is an optional component that is provided as a standalone containerized service for those who do not already operate their own DataEnclave-compatible attestation server.

A key management system (KMS) runs in data owners' trusted environments and only releases keys to DataEnclave CVMs that are verified by the attestation server. A KMS is required, and DataEnclave interoperates with most KMSes that use the standard KMIP protocol. DataEnclave does not provide a KMS.

This architecture eliminates the need to trust the infrastructure operator with unencrypted data. Instead, containers and secrets are only released into an enclave after the party that owns them has verified what that enclave is running through hardware-signed attestation.

The attestation server decides which DataEnclave CVMs receive the keys and secrets they need, but it is designed to keep container images and secrets elsewhere. Container images stay in the VAST Artifact Registry[2] or an external registry, and secrets stay in the data owner's KMS. This allows data owners to choose where their images and secrets are kept and who manages those systems.

Each DataEnclave CVM is trusted by only one party, its owner, and only that owner's keys and secrets are ever released into it. This lets each data owner protect its data with its own keys and policies without placing those keys inside an environment governed by someone else. Other parties' data still reaches the workload, but only as input through the API the workload exposes, such as a prompt sent to a model inference service. The workload's developer is responsible for ensuring that API does not leak data out of the CVM.

DataEnclave CVMs are not tied to a specific attestation server, and data owners can use the DataEnclave attestation server or one they already operate. CVMs can prove their state to Trustee, the open-source attestation server from the CNCF Confidential Containers project, and to other attestation servers compatible with it. VAST also supports proprietary attestation servers. This lets data owners choose an attestation server that runs wherever their security requirements dictate.

DataEnclave is consistent with the NVIDIA Confidential Containers Reference Architecture for running confidential workloads on NVIDIA GPUs. It is deployed and managed through VAST DataEngine and integrates natively with VAST data platform services such as observability and audit.

[2]The VAST Artifact Registry is a new feature that will be co-released with VAST DataEnclave.

Worked Example: Closed Models on Customer Infrastructure

The easiest way to explain how DataEnclave protects confidential workloads and data from third parties is with a fictional example.

Consider the case of Union Capital, a financial services company with sensitive financial data that it would like to analyze using the latest closed-weight frontier models. These models are trained by OpenBrain, a leading AI lab, whose model weights are its most valuable asset. Data governance policy prevents Union Capital from using OpenBrain's hosted inference API, because customer data cannot leave Union Capital's premises. However, OpenBrain is unwilling to deploy their models into third-party environments for concern that their model weights would be stolen and leaked.

DataEnclave provides the service infrastructure that allows OpenBrain to deploy its models into Union Capital's GPU environment without ever exposing OpenBrain's model weights or the proprietary code to Union Capital's administrators.

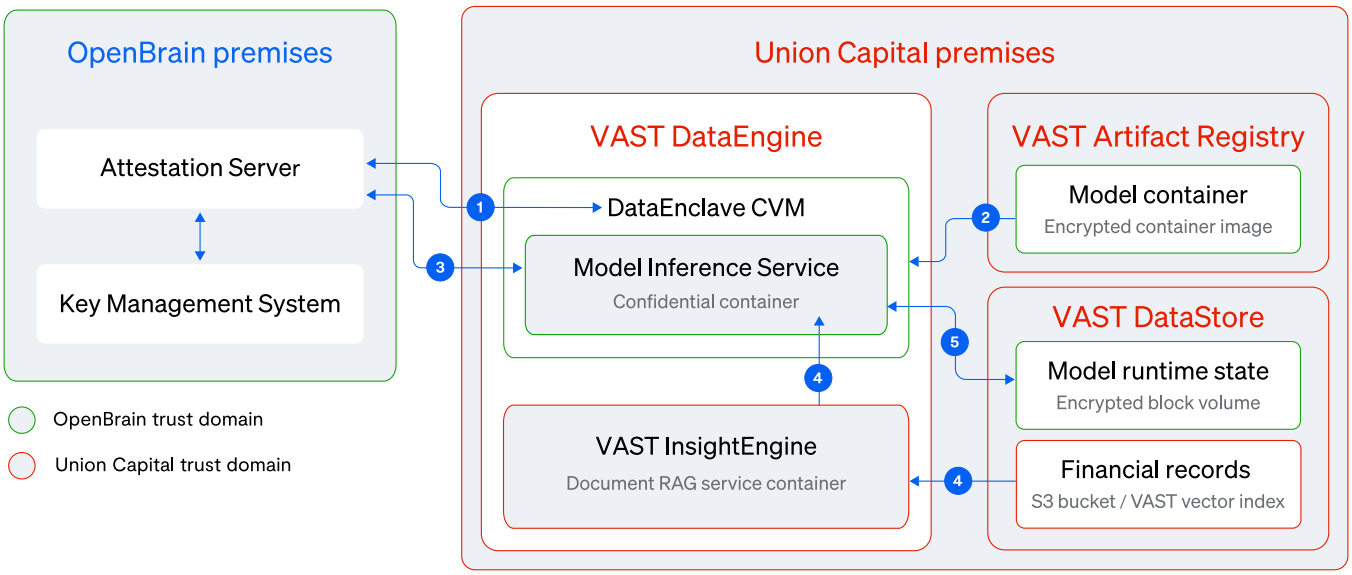


Figure 2 Deploying a confidential model into an untrusted customer environment.

DataEnclave gives Union Capital the ability to create a hardware-isolated, encrypted enclave (the DataEnclave CVM within its GPU environment that is governed by OpenBrain's security policies. From Union Capital's perspective, deploying a confidential AI container or app is no different than deploying any other third-party app through DataEngine with two exceptions: the app manifest contains an explicit enclave agent policy that governs what the untrusted host node can and cannot do inside the enclave, and the GPUs on target nodes must be running in Confidential Computing mode. But to OpenBrain, DataEnclave provides guarantees that Union Capital cannot inspect its container's contents at any step of the container's lifecycle.

Step 1: Proving What is Running Before Anything is Released

Before deploying an OpenBrain container, the DataEnclave Runtime first creates a DataEnclave CVM that uses the CPU's and GPU's memory encryption and protection capabilities. With this hardware isolation enabled across CPU memory, GPU memory, and the interconnect between them, the contents of CPU and GPU memory owned by the CVM are never accessible from outside the CVM.

The CVM then produces a report that attests the CVM's hardware and firmware versions, the runtime agent policies it will enforce, and other details of the CVM's configuration. To prevent forgery, the report is generated and signed by a hardware attestation key stored within

the CPU and GPU hardware roots of trust, and it is bound to a nonce, a single-use value that OpenBrain issues for that specific attestation request. The CPU and GPU vendors own the certificate chains these keys derive from, so an attestation server running in OpenBrain's own trusted infrastructure can confirm the report's authenticity and freshness without ever having to trust Union Capital.

The CVM then sends this report to OpenBrain's attestation server over standard HTTPS. The CVM initiates this exchange, so it requires only outbound connectivity from Union Capital's environment; OpenBrain never needs access into it. The attestation server then verifies that the CVM's hardware, firmware, and policy configurations meet OpenBrain's security requirements. For example, OpenBrain may require that the CVM prohibits the host from exec'ing into its containers and was not launched in a debuggable state.

Step 2: Delivering the Model Without Exposing It

Once OpenBrain has verified the CVM's report, DataEnclave must pull OpenBrain's container directly into the CVM. The confidential container image, which contains the model weights, remains encrypted within the VAST Artifact Registry on Union Capital's VAST Cluster. To ensure Union Capital can never inspect the image or tamper with its contents, the keys that decrypt it and establish its authenticity are held by OpenBrain within their own trusted infrastructure.

As a result, the CVM must present valid attestation to OpenBrain before the image keys are released into it, allowing it to decrypt the image layers and start OpenBrain's inference service. This verify-before-decrypt pattern, where OpenBrain releases keys only into a CVM that has passed attestation, ensures Union Capital never sees any of OpenBrain's confidential data in plaintext.

Step 3: Decrypting the Model Weights

Once running, OpenBrain's inference service decrypts the proprietary model weights file contained in its image, using a second key that is separate from the one that decrypted the image itself. As with the secrets required to decrypt and verify the container image in Step 2, this decryption key for model weights is only released by OpenBrain into the CVM after the CVM's attestation has been verified.

The inference service can use this key to decrypt the contents of the model weights file directly into the CPU or GPU memory owned by the CVM. Because CVM memory is all protected from its host, the decrypted weights are never accessible in an unencrypted state outside of the CVM, making them inaccessible to Union Capital's administrators. And because the DataEnclave CVM's agent policies prevent Union Capital's host from running arbitrary commands inside the CVM, there is no way for the host to extract the decrypted model weights from the running container.

Step 4: Using the Model with Con idential Business Data

OpenBrain's inference service is now running in a DataEnclave CVM within Union Capital's environment, and OpenBrain's model weights are decrypted in GPU memory. The only way for Union Capital to interact with the inference service is through the REST API it exposes.

At this point, Union Capital can treat this locally running OpenBrain inference service just as if it was running in OpenBrain's cloud infrastructure. The difference is that Union Capital never has to send data outside of its network when its agents, chatbots, or other AI-enabled services need to use OpenBrain models.

For example, consider the case where a Union Capital analyst asks which of its loans are most exposed to a downturn in commercial real estate. This would involve sending a query to VAST InsightEngine's semantic search API, which would in turn trigger a series of vector search queries against a VAST vector index and chunk retrievals from a VAST DataBase table to create a set of relevant excerpts drawn from the bank's loan files, internal credit memos, and property valuations. These excerpts would then be included as context within a prompt that is submitted to OpenBrain's inference service running in the DataEnclave CVM.

Up to this point, semantic search, retrieval, and context assembly all happen within Union Capital's own infrastructure. The only Union Capital data OpenBrain's inference service ever sees is the final prompt, and because it also runs inside Union Capital's infrastructure, standard network egress controls can prevent it from sending the prompt anywhere else.

Step 5: Using Confidential Storage for Model Runtime State

While much of this work happens exclusively in CPU and GPU memory, production inference stacks also keep state on persistent storage: KV caches that let the model avoid redoing work across similar requests, precompiled kernels, and, for an agentic service, scratchpads and intermediate results that persist across steps. This state is derived from both OpenBrain's model and Union Capital's prompts, so it is sensitive to both parties. To address these needs, DataEnclave also supports both ephemeral and persistent encrypted volumes.

An ephemeral volume is attached to the confidential container the way it would be for any other container, but its encryption key is generated inside the CVM and never leaves it. Union Capital's infrastructure holds the encrypted blocks and never holds the key, so what is persisted on the volume is unreadable to Union Capital. When the container exits, the key is destroyed, rendering the data on those blocks unrecoverable.

This same mechanism supports persistent volumes when state needs to outlive a single container. In these cases, the volume's encryption key is held by OpenBrain and only released after attestation, again adhering to the principle of verify-before-decrypt. All other data on Union Capital's infrastructure remains protected by the standard access controls and encryption of the VAST AI Operating System. Even where OpenBrain's containers can reach Union Capital's network, they hold neither the credentials nor the encryption keys needed to read data they were not explicitly given.

Attestation, Verification, and Failure

DataEnclave is designed to assume that the nodes on which containers run are not trustworthy by default. Instead, trust of a DataEnclave CVM is only established through attestation and verification.

Strictly speaking, attestation is triggered by a DataEnclave CVM's need for a protected resource, such as a decryption key, from the attestation server as part of the verify-before-decrypt process. Attestation produces a short-lived token, and the CVM presents that token when it requests a resource, so a burst of requests within the token's lifetime does not require repeated attestation. The process is designed to ensure that these protected resources are delivered only into the CVM's protected memory, and every step between the trusted attestation server and the CVM's protected memory is encrypted using an ephemeral key that is private to the CVM.

Attestation Sequence

DataEnclave's process of attestation and verification follows the IETF remote attestation procedures (RATS) architecture (RFC 9334) and is depicted below.

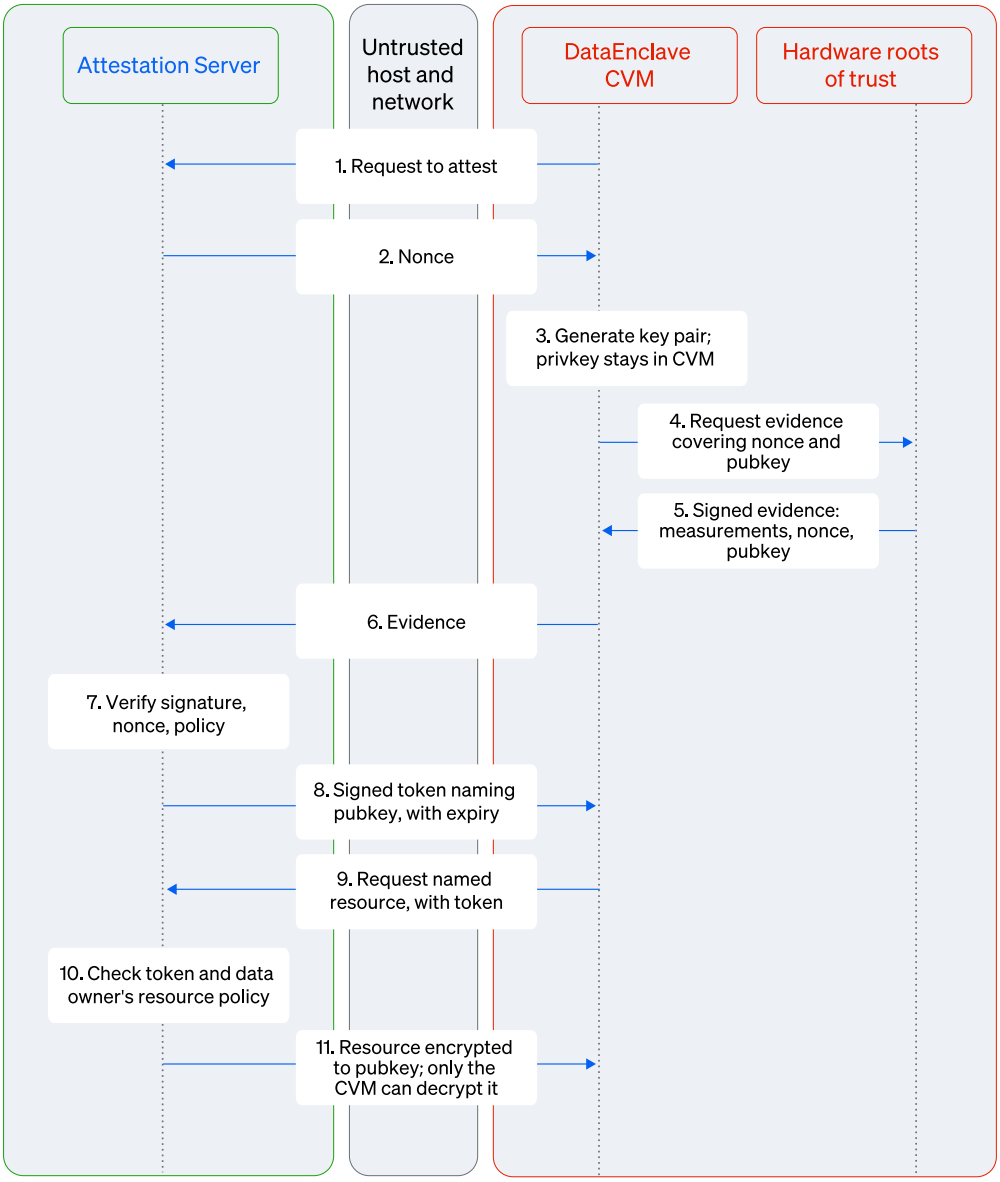


Figure 3 Attestation and resource release sequence.

Once the DataEnclave CVM knows that it needs a protected resource, it begins this attestation process:

1. The CVM asks the attestation server to begin attestation.
2. The server returns a fresh nonce, so an earlier attestation can't be replayed.
3. The CVM generates a key pair whose private key never leaves the CVM.
4. The CVM asks the hardware for evidence and includes the nonce and public key.
5. The hardware returns evidence of the CVM's measurements, the nonce, and the public key in the form of an attestation report, signed with hardware attestation keys certified by the CPU and GPU vendors.
6. The CVM sends the attestation report to the server.
7. The server confirms the evidence in the report
    - is genuine (signed by keys certified by the hardware vendors)
    - is fresh (bound to the same nonce as Step 2)
    - satisfies the data owner's policy (for example, the CVM's measurements match the data owner's reference values, the measurements it has approved).
8. The server issues a signed, time-limited token naming the CVM's public key.
9. The CVM requests a specific protected resource, presenting the token.
10. The server checks the token and whether the data owner's policy allows this release.
11. The server returns the resource, encrypted with the CVM's public key, so only the CVM can decrypt it.

In this final step, the attestation server returns the resource directly. The DataEnclave attestation server operates this way, but other implementations broker access instead, releasing a credential that the CVM uses to access the KMS itself. Either way, release is gated by the same verified evidence and the resource is decryptable only inside the CVM.

Both the CVM key pair (Step 3) and the attestation token (Step 8) are short-lived: each attestation generates a new key pair and a token that expires five minutes after issuance. This ensures that container creation, which requires many protected resources, is not slowed by repeated attestation, while any change to an owner's policy on the attestation server still applies to new resource requests within minutes.

Together, these steps ensure that no one controlling the infrastructure outside the CVM, including privileged administrators and the host software, can obtain a protected resource. Specifically, this flow protects against:

- **Forged Evidence:** The attestation report is signed with a key certified by the hardware vendor (Step 5), so it cannot be fabricated or altered without the attestation server detecting it (Step 7).
- **Replayed Evidence:** The signed report includes a nonce the attestation server issued for this attestation alone (Steps 2 and 7), so a report that was valid in the past cannot be reused.
- **Unauthorized Software or Host Access:** The report includes measurements of the CVM's firmware, image, and configuration, including the agent policy that limits what the host can do inside the CVM, such as running commands in its containers. The attestation server releases nothing unless these match the owner's reference values (Step 7).
- **Key Substitution:** The CVM's public key is part of the signed report (Step 5), so an attacker cannot pair a genuine report with a key of its own.
- **Secret Interception:** Resources are encrypted to that public key (Step 11), and the matching private key never leaves the CVM, so anyone who intercepts the response or steals the token gets only encrypted data.

Attestation Failure or Release Denial

If attestation fails, or if a resource request is denied due to a data owner's policy, the attestation server withholds the resource and logs the denied request. Because decrypting data within a CVM requires key release (and therefore attestation), the workload cannot decrypt or use the model or data, and a workload packaged as an encrypted container image cannot start at all. An unreachable attestation server has the same effect: no keys are released, so new workloads cannot start.

The behavior of a failed attestation or resource release after a container is already running is defined by the application, not DataEnclave. If an application never requests protected resources after it launches, it will never re-attest, and it retains any secrets it was granted at creation. DataEnclave does not govern what happens inside containers, so it is the responsibility of application developers to periodically re-request a protected resource if periodic re-attestation is required. DataEnclave exposes a lightweight REST API inside containers, accessible via loopback, that applications can use to request an attestation token (Steps 1–8) or protected resource (Steps 1–11) and force re-attestation. DataEnclave runs this service inside the CVM to ensure that only its confidential containers can access it.

Trust Assumptions and Limitations

The protections above rest on a small set of trust assumptions:

- **Attestation Server:** The attestation server and the key management system behind it must run in an environment trusted by the data owner, and they are only as effective as the reference values and policies governing protected resources and attestation defined by the data owner.
- **Flaws in the DataEnclave CVM Images:** Attestation verifies the CVM image is authentic and unmodified, but it cannot guarantee it is free of exploitable flaws.
- **Hardware Roots of Trust Within Nodes:** These components are the foundational trust assumption on which the entire confidential computing model is built, so DataEnclave inherits their limitations, including firmware vulnerabilities, side-channel attacks, physical attacks on the server, and compromise of the hardware vendors' signing keys.

In addition, DataEnclave does not protect against:

- **Vulnerabilities in the Workload Itself:** DataEnclave governs what containers can run, but it does not govern what happens inside of them. A workload's API is the one channel designed to cross trust domains, so attacks through it, such as prompt injection that coaxes a model into revealing retrieved data or systematic queries that attempt to reconstruct a model's behavior, are outside what DataEnclave can prevent.
- **Deliberate Exfiltration by the Workload:** DataEnclave protects workloads from the infrastructure, but it does not protect data owners from the workloads they choose to run. A container may be configured to log its API activity to a remote server, and those logs may contain prompts or other sensitive data provided to the workload. The enterprise, as the infrastructure operator, is responsible for securing the network perimeter surrounding these containers. The parties may also consider packaging approaches that make a workload's code verifiable without exposing the model, such as reproducibly built images with separately delivered weights.
- **Loss of Availability**: The infrastructure operator can always decline to run a workload or disrupt one that is running. Confidential computing protects the confidentiality and integrity of data in use, not its availability.

Deployment and Use

The VAST DataEnclave relies on three components.

- The DataEnclave Runtime, which runs on nodes that will run confidential AI workloads. This is an integrated feature of the VAST DataEngine running on a VAST Cluster.
- An attestation server, which runs in a separate, trusted domain external to the VAST Cluster. This can be the DataEnclave attestation server, which is provided via standard Kubernetes packaging including a Helm chart, or an existing attestation server that uses the CNCF Trustee KBS protocol. DataEnclave also supports proprietary attestation servers.
- A Key Management System (KMS), which runs external to the VAST Cluster. VAST does not provide this, and supported KMSes are dictated by the choice of attestation server. The DataEnclave attestation server supports the same KMSes as the VAST AI Operating System which includes many KMIP-based and proprietary solutions.

While the DataEnclave Runtime is always provided by VAST, the choice of attestation server and KMS (and where they are deployed) result in several different modes of deployment:

- Bring Your Own Trust: DataEnclave provides the DataEnclave Runtime only, and the attestation server and KMS are external. This is a typical model for deploying proprietary models in enterprise environments.
- Bring Your Own KMS: DataEnclave provides both the DataEnclave Runtime and DataEnclave attestation server. The DataEnclave attestation server runs in its own enclave, separate from the VAST Cluster. This model is suitable for deployment in untrusted third-party infrastructure such as clouds.
- Air-gapped: An extension of Bring Your Own KMS where keys are held in a hardware security module, and all certificate chains, revocation data, and reference values for verification are cached locally.

Regardless of deployment mode, the process of deploying and managing DataEnclave and confidential AI workloads is similar.

Deploying DataEnclave

Because DataEnclave is integrated into DataEngine, no special installation of the DataEnclave Runtime is required. However, initial activation is necessary to establish mutual awareness between the VAST Cluster managing confidential workloads and the data owner's attestation server:

Registry Configuration: The VAST Cluster must be configured to pull encrypted artifacts from an artifact registry and be given an access token provided by the registry owner. That registry may be external to the VAST Cluster and hosted within the data owner's trust domain, or it can be VAST's own embedded artifact registry. This registry becomes the backing store for the VAST application store described in the next section.

Attestation Server Configuration: The nodes on which confidential AI workloads will run must be registered with the attestation server. This occurs through a dedicated API, authenticating with preprovisioned credentials signed by VAST and scoped to the specific VAST Cluster being configured. This allows data owners to strictly control which nodes in the untrusted environment are allowed to run DataEnclave CVMs and confidential workloads, and data owners can revoke entitlement at any time.

DataEnclave may also be deployed as a turnkey solution that comes pre-loaded with proprietary models from partner providers in the embedded VAST Artifact Registry. In air-gapped installations, DataEnclave also includes an attestation server and tamper-resistant KMS that normally run in the data owner's trusted infrastructure. When deployed this way, all DataEnclave components are preconfigured to serve the pre-loaded models.

Deploying Workloads with DataEnclave

DataEnclave runs always-on containerized services called VAST Applications and handles deploying, scaling, restarting, and updating them. A VAST Application is packaged with everything needed to run them (the container images, the platform services it depends on and the versions it needs, and the variables it expects) and can be deployed via either API or a visual interface.

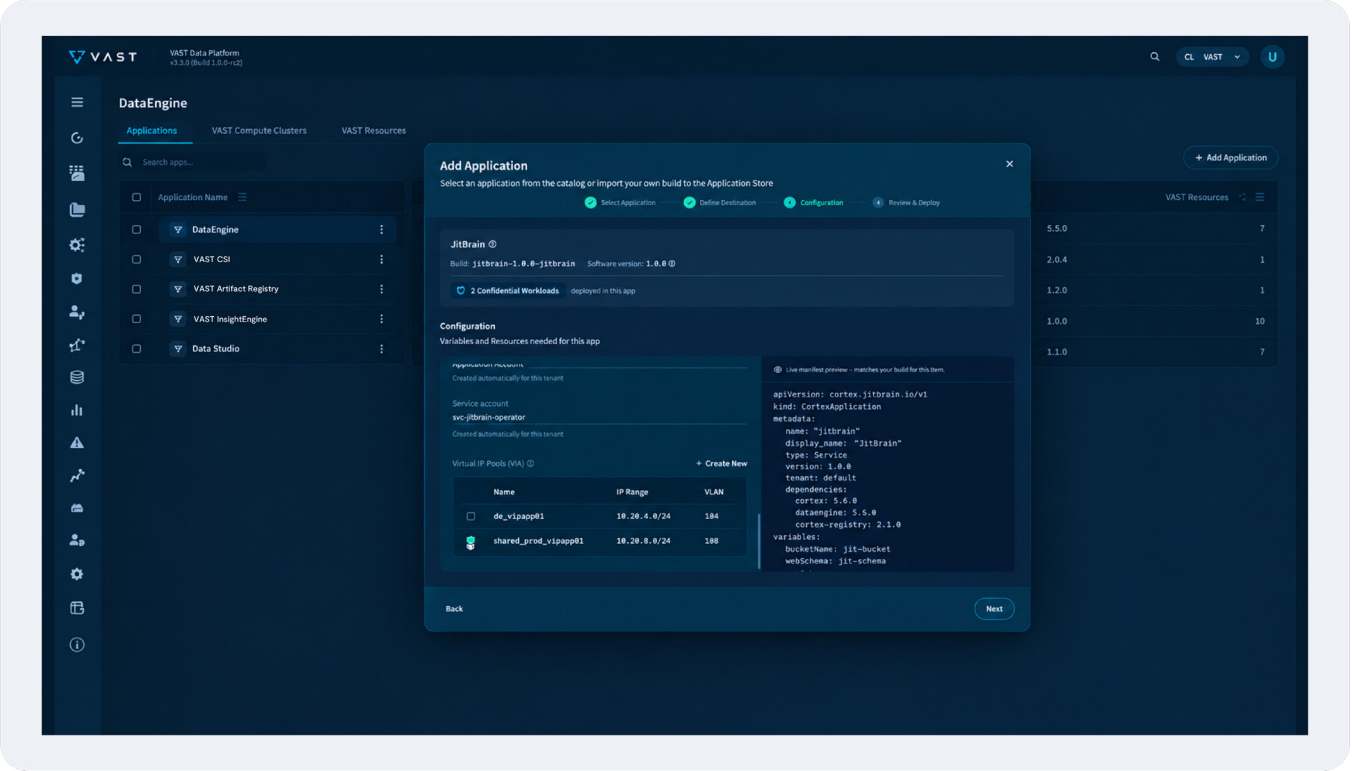


Figure 4 Configuring a VAST Application for deployment including the confidential workload it declares.

Deployment requires first selecting the application and version from the VAST application store, then giving it a destination: an instance name, and the compute cluster (VAST-managed or external Kubernetes) and namespace where it runs. The configuration step collects the variables the application declares, such as the bucket and schema it reads from and the policies that govern access. Before anything is created, the platform checks that the services the application depends on are present at compatible versions. In addition, workloads that require unsupported GPU configurations will not deploy, though DataEnclave supports the full confidential computing capabilities of the underlying GPU platform.

An application's store entry declares which of its workloads are confidential and must be deployed in DataEnclave CVMs. These confidential applications require the address of their attestation servers at deployment time and go through the attestation process as part of container creation as described earlier. If attestation fails or the application is not entitled to run on the selected destination, deployment fails.

VAST Applications that are not confidential run using the standard DataEngine runtime.

Ongoing Management

VAST Applications running in DataEnclave are subject to three different policies, all defined by the data owner who provides the confidential container and attestation server:

- Agent policy, which is enforced by DataEnclave agents inside each CVM. This is defined in the VAST Application's manifest and defines what actions the untrusted host can take inside the CVM (and therefore the confidential workload). The agent policy's digest is bound into the CVM's hardware evidence, so the attestation policy (below) can verify that the CVM launched with the policy the data owner intended.
- Attestation policy, which is enforced by the attestation server. This defines the criteria required to verify a CVM's attestation evidence and issue an attestation token.
- Resource policy, which is enforced by the attestation server. This determines whether a requestor is allowed to access a specific resource (such as a key).

The agent policy is expressed in the OPA Rego language, and when using the DataEnclave attestation server, the attestation and resource policies are as well.

The data owner is responsible for maintaining these policies as the computing environment changes. Updating the agent policy requires updating the VAST Application's manifest, registering a new version of the application with the VAST Cluster, and registering that policy's digest as a reference value in the attestation server. The attestation and resource policies are defined within the data owner's trust domain and are updated using whatever processes are supported by the attestation server implementation.

DataEnclave can pass license keys or other entitlements required to launch confidential workloads into the CVM, but these entitlements must be provided during deployment time and are not provided by VAST. Whether an entitlement is valid, and therefore whether a confidential workload is permitted to run, is handled through attestation and the configured resource policy. This is the responsibility of the data owner to configure and maintain.

DataEnclave CVM images are built, hardened, and signed by VAST. They carry a minimal package set and do not support interactive login, reducing the attack surface inside the CVM to the workload itself. Each image produces a set of hardware measurements recorded when the CVM launches, identifying the firmware and kernel the CVM started with. These measurements appear in the same hardware-signed attestation report that carries the agent policy's digest and the other values the attestation server verifies.

Those measurements are how the data owner retains control despite not building the CVM image themselves. The data owner registers the measurements they accept in the attestation server, and a CVM launched from any other image fails attestation and the workload cannot run. Updating a CVM image is therefore a two-party change: a new image produces new measurements, so the data owner must register these measurements as reference values before deployments can begin using the image.

Audit and Evidence

Because DataEnclave is built on the VAST AI Operating System, it has access to the same scalable, queryable, tamper-proof audit logging available to DataEngine. DataEnclave records workload lifecycle activity in the VAST DataBase: which application and version launched, from which CVM image, on which node, at what time, and under which deployment configuration. Security and compliance teams can query what ran and where without exposing the data or models protected inside the confidential environment, since the log records the identity of what ran rather than anything it processed.

The data owner's record of activity is generated by the attestation server running in their trusted infrastructure. When the DataEnclave attestation server is being used and is deployed as a VAST Application in the data owner's environment, it also gains the audit capabilities of DataEngine and records which evidence was verified against which policy, whether it passed, and which resources were released to which attested workload. Changes to reference values and policies themselves are recorded and auditable as well.

Removing the Trust Problem

The trust problem described at the start of this document has historically been managed with legal contracts: a model provider licenses its weights under terms that prohibit extraction, and an enterprise gets written assurances that its prompts will not be retained. Contracts work when a failure can be remedied afterward, but a leaked set of frontier model weights or a copied corpus of regulated customer data cannot be. DataEnclave addresses this by preventing the exposure using hardware-enforced confidential computing. Nothing is decrypted until hardware that the infrastructure operator cannot forge has proven what is running, and that proof is verified by an attestation server the data owner controls.

For enterprises, DataEnclave allows the model to come to the data rather than the reverse. Semantic search, retrieval, and prompt assembly happen entirely within infrastructure the enterprise controls, and the only thing that crosses into the model provider's container is the assembled prompt. Because that container runs inside the enterprise's own network, the enterprise governs what it can reach.

For model providers, DataEnclave allows a model to be deployed into an environment the provider does not control. Weights and other secrets are decrypted only into memory the provider has attested under a policy the provider wrote and enforces from its own attestation server. Entitlement is revocable at any time and without the customer's cooperation, so a customer whose governance prohibits hosted inference becomes reachable without the model leaving the provider's control.

For infrastructure providers, DataEnclave removes the operator from the trust equation. A confidential AI workload can run on hardware the operator administers without the operator being able to see the model or the data inside it. The operator is responsible for availability and the integrity of the platform and is never responsible for safeguarding another party's data. The audit log still shows which application and version ran, on which node, at what time, and under which deployment configuration, none of which reveals any data the workload processed.

No party has to place its keys inside an environment governed by another. Each defines the security baseline it requires, and each verifies that baseline has been met through hardware evidence it can appraise itself.

©2026 VAST Data, Inc. All rights reserved. All trademarks belong to their respective owners.

09/26 I 15002