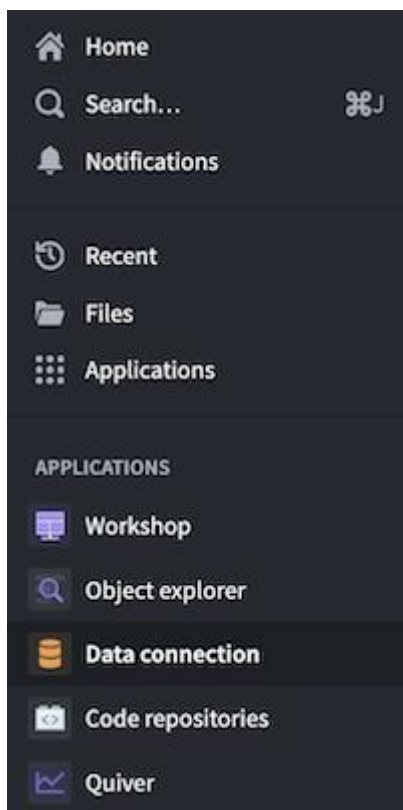


Connect to data

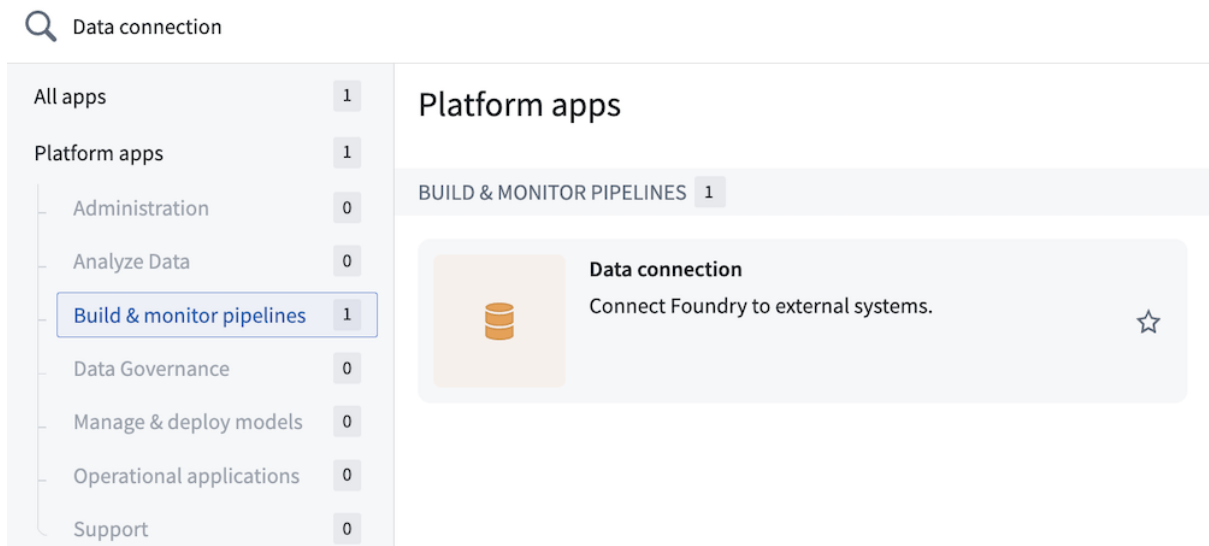
Data Connection

Data Connection is an application used to synchronize data from external systems for use in Foundry. Users can use Data Connection to sync data into Foundry for use in the data integration, modeling, and Ontology layers. Additionally, Data Connection enables setting up outbound connections to enable writeback to external systems via Webhooks and data exports.

Access the Data Connection application by selecting the icon from the workspace navigation bar.



You can also find the Data Connection application by searching in the Applications Portal.



Foundry standardizes the data connection process with the following three principles:

- Robustness
- Extensibility
- Ease of use

Robustness

Often, connecting data between systems is prone to failures that can be difficult to recover from. Issues out of the user's control in the external environment (e.g., poor network connectivity, disk failures, or unresponsive source systems) can affect data syncs, breaking downstream analytics pipelines as well. Incomplete or corrupt data is not only a technical challenge, but also potentially dangerous for the organization if it is used unnoticed or not available for urgent needs.

Foundry proactively addresses these common failure points with automatic retries upon failures, use of simple functions (e.g., filesystem and database syncs) to pull data in small batches with low complexity queries from the source systems, and an integrated data health monitoring system to alert on critical failures and surface other pipeline health issues. Combined, these features minimize the risk of incomplete or corrupt data.

Foundry is also distinguished by a philosophy that data should be ingested "as-is" from its most raw source, with no external preprocessing. In the absence of external preprocessing, the branched and version-controlled Foundry pipeline becomes the single source for all of the changes that happened to raw data on its journey to Ontology, and any issues that arise on that path can be identified and resolved inside the platform. Data Connection adheres to this design philosophy by supporting both tabular and file-based syncs and deliberately offering minimal options for transforming

the data before it arrives in the destination dataset (the starting point of the Foundry pipeline).

Extensibility

Enterprises have a diverse, complex array of systems that add value individually and as an integrated system. Each system has its own requirements for integration and some systems require unique capabilities or features that would make it typically difficult to integrate.

Foundry provides out-of-the-box integration with well-known system types (e.g., relational databases, FTPS, HDFS, S3, SFTP, and local directories) as well as the flexibility to connect and sync data from new system types. In many cases, new systems can reuse an existing plugin or one with just minor changes. Core features (e.g., scheduling and uploading) are standardized so that only the connection itself needs to be adjusted.

Ease of use

Managing data connections between systems can be an involved process that puts a significant burden on administrators who are responsible for every step: syncing, authentication, scheduling and orchestration, and monitoring.

Foundry abstracts away this complexity with backend services that take on the bulk of the work as users set up and manage pipelines through a simple frontend user interface. This lowers barrier of entry to a typically complex technical task, making it possible for more users to perform data connection.

Core concepts

Sources

A **Source** represents a single connection, including any configuration necessary to specify the target system and the credentials required to successfully authenticate. Sources must be configured with a particular runtime depending on the networking between the Palantir platform and the target system. The runtime also defines where any capabilities used with the source will be run.

A source is set up based on a particular **connector** (also referred to as a **source type**). A broad range of connectors are available in the Palantir platform, designed to support the most common data systems across organizations. Depending on the connector and runtime selected, different capabilities may be available.

For systems without a dedicated connector, the generic connector or REST API source may be used with code-based connectivity options such as external transforms, external functions, and compute modules.

Credentials

A *credential* is a secret value that is required to access a particular system; that is, credentials are used for authentication. Credentials can be passwords, tokens, API keys, or other secret values. In the Palantir platform, all credentials are encrypted and stored securely. Depending on the runtime, secrets may be stored locally on a Data Connection agent, or directly in the platform.

Some sources are able to authenticate without storing any secrets, such as when using OpenID connect, outbound applications, or a cloud identity.

Runtimes

Sources must be configured with a **runtime**. The runtime defines the networking configuration and where capabilities are executed.

Palantir allows you to use three different runtimes to connect to your systems. In general, if the system to which you're connecting can accept inbound connections from the network where your Foundry instance is hosted, you should use a direct connection runtime. If this is not possible, and the source type you are using supports an agent proxy, this is the preferred agent-based option. If neither of the other runtimes are available, you should fall back to using an agent worker.

Not all runtimes are available for all source types.

Runtime option	Networking	Capability execution
Direct connection [recommended]	The target system must allow direct inbound traffic from Palantir; for standard Foundry instances, this usually means allowing inbound traffic from the standard egress IP addresses viewable in Control Panel or the Data Connection application.	Capabilities are executed in Foundry.
Agent proxy	An agent installed on your infrastructure is used to reverse proxy traffic to systems that are not reachable through a direct connection.	Capabilities are executed in Foundry.
Agent worker	An agent installed on your infrastructure is used to run jobs that interact with your target system and separately push or pull data from Foundry.	Capabilities are executed on a customer-provided Linux host.

Direct connection

Direct connections enable users to connect to data sources accessible over the Internet without needing to set up an agent. This is the preferred source connection method if the data source is accessible over the Internet; it avoids the operational overhead of setting up and maintaining agents and offers high uptime and performance.

When using direct connections with a Foundry instance hosted on-premise, the target system must be reachable from the network where your Foundry instance is running. If this is not the case, you must use one of the agent-based runtime options.

Agents

An **agent** is a piece of software provided by Palantir that runs on a host within your network. The agent connects to your source systems, and can also communicate with Foundry. An agent is required in order to use the **agent proxy** and **agent worker** runtimes. The same agent may be used as an agent proxy or agent worker, which is determined when using the agent with a particular source.

Agent proxy runtime

The agent proxy runtime is used to connect to data sources not accessible over the Internet. The agent acts as an inverting network proxy, forwarding network traffic originating in Foundry into the network where the agent is deployed, and relaying traffic back to Foundry. This allows capabilities in Foundry to work almost exactly the same as when using a direct connection but without requiring you to allow inbound network traffic to your systems originating from Foundry's IP addresses.

For high availability, multiple agents can be configured with non-overlapping maintenance windows to ensure there is always an active agent to proxy connections to the target systems reachable through the agent proxy.

Agent worker runtime

The agent worker runtime is used to connect to data sources not accessible over the Internet. An agent worker should only be used when the desired connector does not support the *agent proxy* runtime. Agent worker runtimes are associated with a single or multiple agents that store the source configuration and credentials locally in an encrypted format, and run source capabilities on the agent itself.

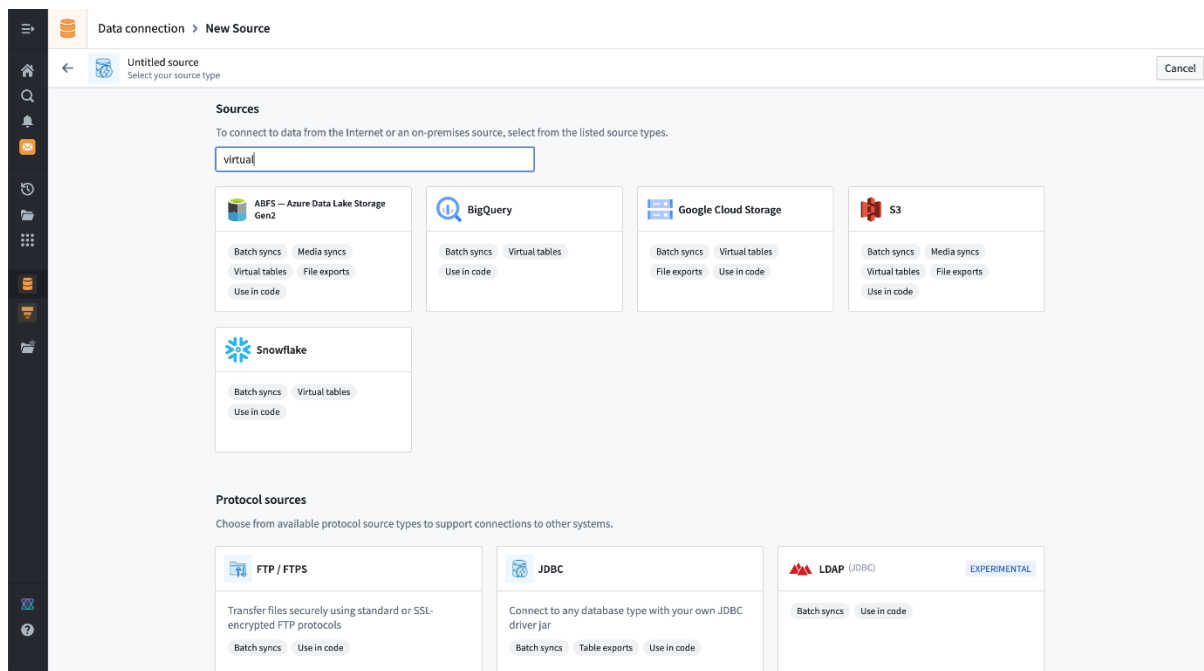
Capabilities

Sources may support a variety of *capabilities*, where each capability represents some functionality that can run over the source connection. There are a wide range of supported capabilities for bringing data into Foundry, pushing data out of Foundry, virtualizing data stored outside of Foundry, and making interactive requests to other systems.

Capability	Description
Batch syncs	Sync data from an external source to a dataset.
Streaming syncs	Sync data from an external message queue to a stream.
Change data capture (CDC) syncs	Sync data from a database to a stream with CDC metadata.
Media syncs	Sync data from an external source to a media set.
HyperAuto	Sync an entire system automatically.
File exports	Push data as files from a dataset to an external system.
Table exports	Push data with a schema from a dataset to an external database.
Streaming exports	Push data from a stream to an external message queue.
Webhooks	Make structured requests to an external system interactively.
Virtual tables	Register data from an external data warehouse to use as a virtual table.
Virtual media	Register unstructured media from an external system as a media set.
Exploration	Interactively explore the data and schema of an external system before using other capabilities.
Use in code	Use a source in code to extend or customize any functionality not covered by the point-and-click-configurable capabilities listed above.

Additional capabilities are being developed, and capability coverage is regularly updated in the documentation for specific connectors.

Supported capabilities for specific connectors are also displayed on the new source page in the Data Connection application. It is possible to search both by connector name and by capability. The example below shows the results of a search for sources that support a "virtual" option.



Batch syncs

Batch syncs read data from an external system and write it into a Foundry dataset. A batch sync defines what data should be read and which dataset to output into in Foundry. Batch syncs can be configured to sync data incrementally and allow syncing data both with and without a corresponding schema.

In general, there are two main types of batch syncs:

- **File batch syncs:** Allows syncing files without a schema directly into a Foundry dataset. These files may then be accessed in downstream transforms; for example, using file-based transforms. The most common systems that support file batch syncs are filesystems and blob stores such as S3, ABFS, Google Cloud Storage, SMB, and Sharepoint online.
- **Table batch syncs:** Allows syncing data with a schema into a Foundry dataset. Part of the sync definition in this case also includes how to translate types between the external system schema and the supported Foundry schema options. The most common systems that support table batch syncs are databases and SaaS providers such as Microsoft SQL Server, Postgres, SAP, Salesforce, and Netsuite.

Streaming syncs

Streaming syncs provide the ability to stream data from systems that provide low latency data feeds. Data is delivered into a streaming dataset. Some examples of systems that support streaming syncs include Kafka, Amazon Kinesis, and Google Pub/Sub.

Change data capture syncs

Change data capture (CDC) syncs are similar to streaming syncs, with additional changelog metadata automatically propagated to the streaming dataset where data is delivered. This type of sync is normally used for databases that support some form of low-latency replication.

Media syncs

Media syncs allow importing media data into a media set. Media sets provide better tooling than standard datasets for ingesting, transforming, and consuming media data throughout Foundry. When dealing with PDFs, images, videos, and other media, we recommend using media sets over datasets.

HyperAuto

HyperAuto is a specialized capability that can dynamically discover the schema of your SAP system and automate syncs, pipelines, and creation of a corresponding ontology within Foundry. HyperAuto is currently only supported for SAP.

File exports

File exports are the opposite of file batch syncs. When doing a file export, data is taken directly from the underlying files contained within a Foundry dataset, which are written as-is to a filesystem location in the target system.

Table exports

Table exports are the opposite of table batch syncs. When performing a table export, data is exported as rows from a Foundry dataset with a schema, which are then written to a table in the target system.

Streaming exports

Streaming exports are the opposite of streaming syncs. When doing a streaming export, data is exported from a Foundry stream, and records are written to the specified streaming queue or topic in the target system.

Webhooks

Webhooks represent a request to a source system outside of Foundry. Webhook requests can be flexibly defined in Data Connection to enable a broad range of connections to external systems.

Virtual tables

Virtual tables represent the ability to register tabular data from an external system into a virtual table resource in Foundry.

In addition to registering individual virtual tables, this capability also allows for dynamic discovery and automatic registration of all tables found in an external system.

Virtual media

Virtual media works similarly to media syncs, allowing media from an external system to be used in a media set but without copying the data into Foundry. Instead, media files contained in an external system can be registered as virtual media items in a specific media set.

Exploration

The interactive **exploration** capability allows you to see what data is contained in an external system before performing syncs, exports, or other capabilities that interact with that system.

Exploration is most commonly used to check that a connection is working as intended and that the correct permissions and credentials are being used to connect.

Use in code

The ability to use connections from code is intended to allow developers to extend and customize connections from Foundry to other systems. Palantir's general principle is that anything possible in the platform using dedicated connectors and point-and-click configuration options should also be achievable by writing custom code. At any point, developers should be able to switch to code-based connectivity for more granular control over the functionality or performance of workflows that perform external connections.

Any connector may be used in code; in most cases, we recommend using either the REST API source or generic connector when connecting from code.

Use in code option	Description
	External transforms allow transforms written in Python to communicate with external systems.
External transforms	External transforms are a code-based alternative for file batch syncs, file exports, table batch syncs, table exports, and media syncs. External transforms may also be used to register data into virtual media sets and virtual tables.
External functions (webhooks)	External functions written in TypeScript support importing a source in order to invoke existing webhooks defined on that source. This allows existing webhook calls to be wrapped in custom typescript logic and error handling.
External functions (direct)	External functions now allow direct calls to external systems using fetch for TypeScript and requests for Python. External functions are a code-based alternative for webhooks.

Use in code option	Description
	External functions with direct external calls are not yet generally available.
	Compute modules allow for long-running compute and writing connections in arbitrary languages.
Compute modules	Compute modules may be used as a code-based alternative for streaming syncs, streaming exports, change data capture syncs, and webhooks.
	Using sources in compute modules is not yet generally available.
External models	External models currently do not support importing sources. Instead, you must use network egress policies directly.
Code workspaces	Code workspaces currently do not support importing sources. Instead, you must use network egress policies directly.
Code workbooks	Code workbooks currently do not support external connections.

Not all source configurations allow usage of the **Use in code** capability. Agent worker connections are not supported, and some credential types such as cloud identity, outbound application, and OIDC may not currently be used from code.

Other concepts

Data connection also includes a variety of other concepts relevant for specific workflows. Some concepts were previously used and are now sunset, but are retained here for reference.

Syncs

Historically, the term **Sync** was used in a generic way to refer to bringing data into Foundry. Syncs are now separated into the more specific capabilities listed above. More details are available for each capability, such as **batch syncs**, **streaming syncs**, **change data capture syncs**, **media syncs**, and so on.

Tasks [Sunset]

Sunset

Tasks are in the sunset phase of development and will be deprecated at a future date. Full support remains available. We recommend migrating your workflows to code-based connectivity options.

The plugin framework used to implement connectors allows custom extensions called **tasks**. Tasks represent a unit of functionality configured by providing YAML and implemented in Java as part of the Data Connection plugin. Palantir has stopped developing new tasks, and all officially supported capabilities have migrated away from using tasks.

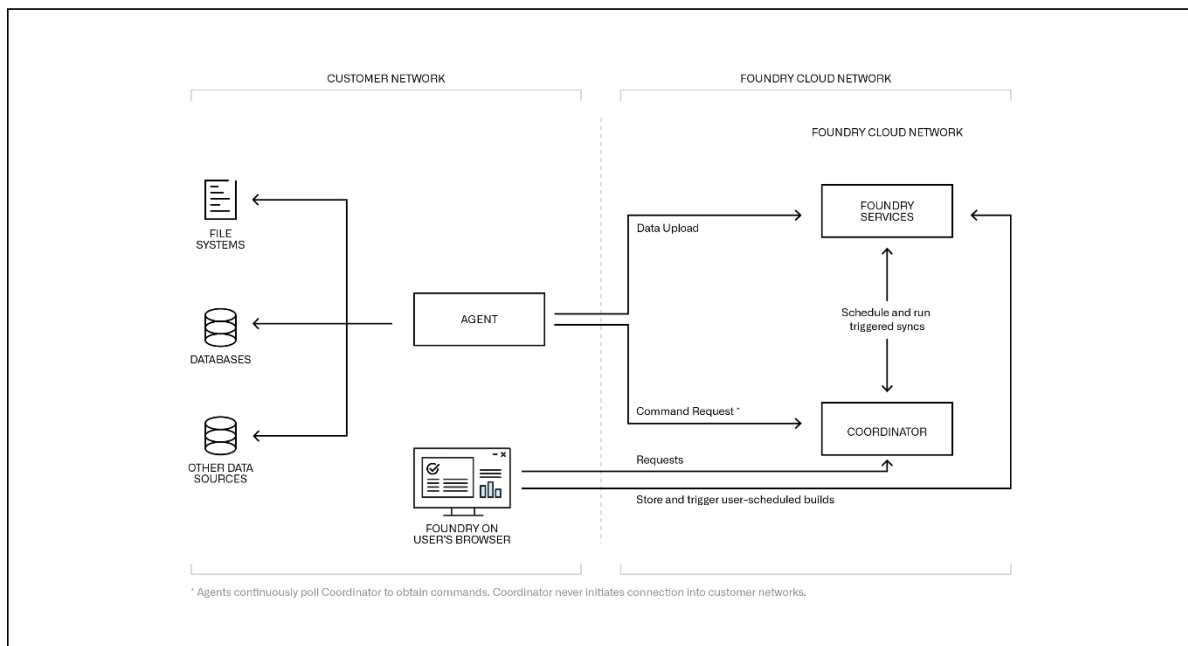
Agent architecture

Data Connection agents are only required when configuring a source runtime to connect to on-premise system. Direct connections are preferred when the data source is accessible via the Internet.

Data Connection agents are a Foundry controlled application that run in a customer network on a customer-controlled host.

- The **Data Connection application** enables authorized users to configure, explore, and run syncs to Foundry.
- The **Agent Worker** is a service that actually performs the data syncs including carrying out operations, running the syncs, uploading data to Foundry, and caching metadata.
- The **Agent Proxy** is a service that provides network connectivity for services in Foundry to connect to source systems in the agent's network.
- The **Coordinator** is responsible for configuring and executing jobs that tell the Agent how to sync data. It sits in the Foundry cloud enclave.

The Agent only communicates with the Coordinator via unidirectional outbound connections secured by HTTPS from the customer network into the Foundry platform. In *agent proxy* mode these outbound connections are used to provide bidirectional network connectivity to services within Foundry.



Initial setup overview

Before starting, it is important to recognize that this first step towards connecting your organizational data to Foundry is fundamentally a **networking concept**. The initial setup is best done by someone familiar with network engineering and aware of the organization's network topology and configurations, such as firewall rules.

Conceptual overview

Connecting data to Foundry requires that the following three components are installed or configured, in this order:

1. **Connection:** Required to access a data source.
 - **Agent:** Connect to software running on your system; required to access private networks and on-premises data sources.
 - **Direct connection:** Connect to a data source over the Internet; preferred when connecting over a public network.
2. **Source / Connector:** Used to access data external to Foundry.
3. **Sync:** Ingest or export data into Foundry.

An agent is a piece of Palantir software that runs within your organization's network. The agent functions as a secure intermediary between your organization's data sources and your Foundry instance. An agent connection is required to access sources running on private networks or on-premises systems. A single running agent can support multiple sources and syncs.

A direct connection is a connection to a data source that is accessible over the Internet, such as a REST API, an SFTP server, or an Azure storage account. You can configure a direct connection to avoid setting up an agent while still receiving excellent uptime and

performance. Direct connections require a network egress policy for your enrollment and connection credentials.

A source, or connector, is any sort of external data system that you connect to Foundry. For example, a source could be a Postgres database, an S3 bucket, a filesystem on a Linux server, an SAP instance, or a REST API on the internet. A configured source is required to establish any syncs to Foundry, and data must be synced from the source into a dataset before it can be used in Foundry.

A sync reads specific data from a source and ingests it into Foundry. For example, if you have a PostgreSQL database source that contains multiple tables, you might configure a sync to ingest one specific table into Foundry. Once a sync has successfully run, the result in Foundry will be a dataset to use across all of Foundry's data pipelining, model development, and analytical tools.

Roles and workflows

Most Foundry users will never need to set up a new agent themselves. Agent setup requires an IT-focused skill set, though the same agent can be reused to support multiple sources and syncs. Some organizations can operate long-term with agents set up during the first week of a Foundry deployment. New agents are only needed to access data that your existing agents cannot access (due to network segmentation or data scale, for example) or to set up an additional agent to allow for high availability.

The table below summarizes the configuration frequency and skill set required for maintaining the resources required for connecting to data:

Resource	Frequency of configuration	Typical user role	Knowledge required
Agent	Rare	IT / Network Engineer	Network and firewall policies; Linux VMs; SSH
Source	Occasional	IT / Network Engineer; Data Engineer	Debugging network access; credential management
Sync	Frequent	Data Engineer; Data Scientist	Writing SQL queries; managing files

High availability

We recommend setting up redundant hardware to establish a high availability (HA) architecture. High availability increases resiliency and allows no-downtime maintenance during operating hours.

Foundry offers HA at the source level, meaning that if a source is assigned to multiple agents, Foundry will dispatch ingestions to one of the healthy agents. We strongly recommend configuring agents in a high availability setup at the start of source

creation; adding extra agents to a created source requires re-entering the credentials for that source.

The following best practices are recommended when setting up high availability:

- Always install agents by pairs, on similar hardware.
- Give each agent in a pair similar names, such as agent-1 and agent-2.
- Systematically assign both agents in a pair to every source.
- Configure non-overlapping upgrade windows on both agents in a pair. Upgrade windows should be during business days and provide sufficient soaking time. Doing so ensures that any unexpected issues with an update will be contained to a single agent and can be detected by operators or administrators.

Set up a direct connection

Direct connections depend on Foundry's container infrastructure which is only available in Foundry's managed SaaS platform. As a result, cloud-based direct connections may not be available in your environment.

If you are trying to connect to a data source which is accessible over the Internet, such as a REST API, an SFTP server, or an Azure storage account, you can configure a direct connection to avoid needing to set up an agent. Using a direct connection has a number of advantages:

- No need to provision, configure, and manage an agent and its host
- Avoids routing Internet-to-Foundry through your network
- Offers excellent uptime and performance as cloud-based Syncs do not depend on an agent software package or its host

If you are interested in configuring a cloud-based direct connection, follow these steps:

1. Configure a network egress policy for your enrollment.
2. Provision credentials to connect to your data source.
3. Create the Source in Data Connection.

Configure a network policy

You must have the *Information security officer* role on your Enrollment to configure network egress. If you do not have permissions to configure egress, contact your Palantir representative for help.

The *Information security officer* role can be found in the Enrollment permissions section of the Control Panel. An administrator needs to have the *Enrollment administrator* role in order to see this section.

To configure a network policy, navigate to Control Panel using the **Other workspaces** link in the Workspace sidebar. In Control Panel, select **Network egress** in the sidebar. If you can't see this option, contact your Palantir representative to go through the following steps.

Add network egress policy

1 Configure egress

2 Configure permissions

Policies must be applied to sources in Data Connection to take effect.

Description
Describe the type of traffic and/or destination that this policy will allow.

Export to cloud storage provider

Address
Specify the address that this policy will allow.

CIDR 192.168.0.0/16

Port number
Specify the port number or range that this policy will allow.

Single port 443

☐ Make this a global policy **Not recommended**

Global policies are applied to all user workflows, including code workbooks, code transforms, and modeling live deployments. This should only be enabled for highly trusted destinations.

Next

Add a network policy by selecting **Add network policy**. Add a description and connection details, similar to the details you provided when contacting Palantir:

- If you are connecting via HTTP(S), add the DNS hostname of your data source
- If you need to use a non-HTTP protocol, add a CIDR address and port

Keep the default **Optional** policy type selection, and select **Add network policy**.

Provision credentials

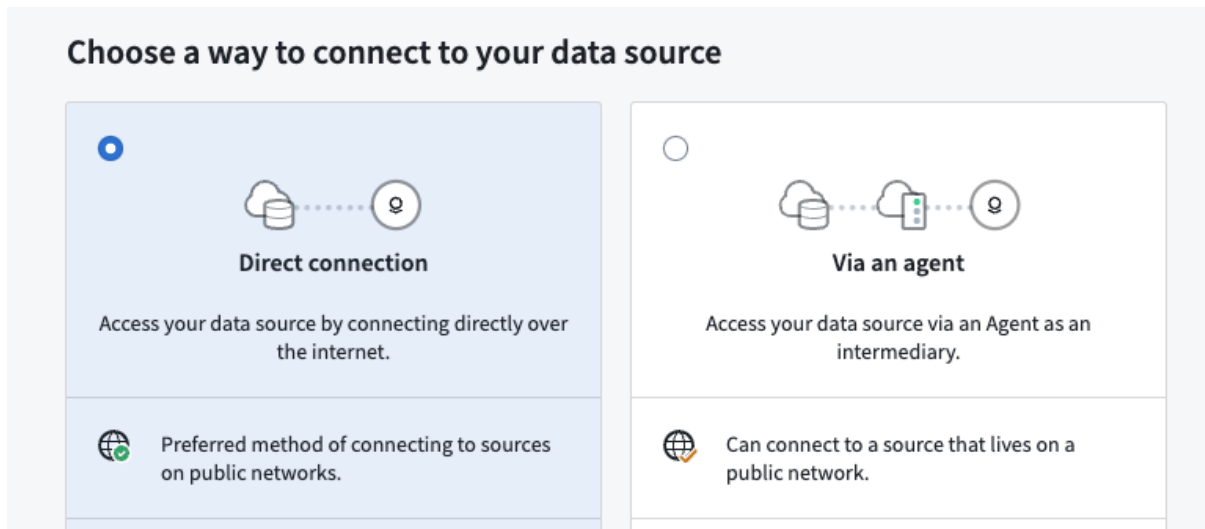
In the majority of cases, Foundry will require authorized credentials (such as a username and password) to access Sources. It is best practice to use a service account specifically for Foundry.

Provision a service account for the Source following any internal guidelines and processes that your organization has for establishing service accounts. Note the credentials before proceeding to the next step.

Create the Source in Data Connection

Once the above steps are done, you can proceed with creating the Source in Data Connection:

- After logging in, navigate to **Data Connection** using the sidebar.
- Select the **Sources** tab.
- Select **New source** in the top-right.
- Select the source type corresponding to your data source.
- Select **Direct connection**, then select **Continue** in the bottom right.



Save the Source in a Project

Next, name your Source and choose a Project to place it in. We generally recommend creating a new Project for each Source, as this provides the cleanest way to permission datasets derived from this Source.


Select **Create source and continue** in the bottom right.

Choose your network policy

On the next page, select the network policy you configured earlier by clicking **Use existing policy** and searching for the policy name.

Network Connectivity


↔ Switch to connect via Agent



Select a policy


A network egress policy is required to connect to sources outside Foundry's secure network. Select an existing policy if one already exists, otherwise create a new policy. [Learn more](#)


+ Use existing policy

 Create new policy


Add egress policies

×


 mydomain

 mydomain.com (Port 443)


Select

 mydomain.com (Port 443)

Select

 mydomain.com (Port 44)

Select

 mydomain.com (Port 443)
No description

Select

Cancel

Select

Configure Source and add drivers

Add details about how to connect to your source. These details will depend on the source type you are using and typically consist of basic credentials such as connection URLs, cloud provider regions, and so on.

JDBC sources may require adding and selecting **drivers** required to connect to your source. Although many drivers ship out-of-the-box with Foundry, you may need to upload and select a driver to proceed.

Add credentials

Add the credentials you provisioned previously to allow the direct connection to connect to your data.

Save and continue

Select **Save** in the bottom right to complete setting up your direct connection. Once your Source is fully set up, you can proceed to set up a Sync to bring data into Foundry.

Set up an agent

An **agent** is a downloadable program installed within your organizational network and managed from Foundry's Data Connection interface. Agents have the ability to connect to different data sources within your organizational network. They are used by both the **agent worker runtime** to read data from those sources and securely ingest to Foundry with a restricted access token, and by the **agent proxy** runtime to provide network connectivity to those sources.

First, complete the following:

1. After logging in to Palantir, navigate to **Data Connection** using the left sidebar.
2. Select the **Agents** tab.
3. Select **New agent** in the upper right corner.

Once you have the agent running and you want to connect a source to Foundry, you must obtain credentials for the source system that the agent can use to securely read data. Depending on your organization's network setup, you may also need to configure network settings to allow the agent to reach the source system.

Review the sections below to start setting up your agent:

Setup

Create agent host

For the agent program to successfully run, it must be hosted in a suitable environment (ideally, an environment using Linux as an operating system).

The most commonly used hosting method for Foundry agents is provisioning a Linux virtual machine (VM) in a cloud environment. For example, you could provision a Linux VM in AWS, Azure, or GCP, but you could also host the agent on a Linux server belonging to your organization. Note that while it is possible to host Foundry agents on Windows, this is not recommended by Palantir and should only be used if it is not possible to host in a Linux environment.

The host you use for the agent should be used exclusively for running a single Foundry agent, not colocated with any other services or processes. Running multiple Foundry agents on the same machine is not supported.

Once you have a suitable location to host your agent, the next step is to ensure the host will meet the necessary hardware and OS requirements for a Foundry agent to work. These requirements include the following:

- **64-bit Linux** or other Linux operating system (recommended RHEL 8, Ubuntu 22.04, or equivalent)
 - Agents run on their own JDK that is compiled for Linux/x86-64. If necessary (for example, when running on AWS Graviton or another ARM-based CPU), it is possible to run an agent on a separate JDK by modifying the value of `javaHome` in `service/bin/launcher-static.yml`.

We generally do not recommend running agents on a separate JDK, and support for this may not be available in the future.

- **4 CPU** cores
- **16 GB** RAM
- **500GB** free disk space mounted at `/opt` (preferably SSD)

The recommended limits are as follows:

- **Core file size:** Hard and soft limit of 0
- **Open files:** Hard and soft limit of 262144
- **Running processes:** Hard and soft limit of 65536
- **Stack size:** Hard and soft limit of 32768
- **Max locked memory:** Hard and soft limit of "unlimited"



A server to provision a host

You will need a Linux server in your network to create an agent

Minimum hardware requirements

Cores	4
Memory (RAM)	16 GB
Storage	500 GB
OS	64-bit Linux

Configure agent network access

Assuming your agent has been installed on a host within your organizational network, the agent will require network egress to reach the Foundry VPC (Virtual Private Cloud) which is accessed through the Internet. If your network does not allow egress by default, this may require a specific

configuration to allow the outbound connection from your agent (and/or its host) to your Foundry instance, such as opening a firewall or configuring a proxy for egress.

As a first step, ensure that egress from your server to Foundry is available. You can copy the domain name and port from the **Server Setup** tab in the agent setup workflow in the platform to appropriately configure your network access.

To validate that your host can communicate with the Foundry VPC, execute the following command on your VM:

```
curl -s https://<your domain name>/magritte-coordinator/api/ping > /dev/null && echo pass || echo fail
```

If everything is working as expected, you should see pass as an output.

Note that a ping indicates an incomplete test of connectivity to the Foundry VPC.

Secure an agent host

To only allow your users to connect to a limited set of destinations within your network, we recommend configuring the firewall of the agent host to block all traffic except to the desired destinations. Be sure to still allow the agent host to talk to Palantir.

Set up automatic restarts

If you do not have automatic restarts set up, you will have outages whenever the agent crashes or the agent host restarts.

To set up automatic restarts for an agent manager if it crashes, run the command `${AGENT_MANAGER_DIR}/service/bin/auto_restart.sh` from the agent manager's service directory on the VM or machine terminal as a user with permission to create cron jobs.

If you need to halt the automatic restarts (when upgrading the agent manager, for example), you can do so by running `${AGENT_MANAGER_DIR}/service/bin/auto_restart.sh clear`.

Save agent resource in a Project

Next, you must give your new agent a name and choose a Project in which to save it. In Foundry, an agent is considered a resource that is saved into a Project to allow for highly configurable permissions.

We recommend creating a new Project in which to store your agent.

Download and install the agent

Once you have your hardware provisioned for your agent, the next step is to download the agent software from Foundry and install it on the host. Extract it, and start the agent.