Connecting to Data Sources

One of the primary reasons Tableau champions advocate for Tableau Server is the collaboration that it enables. As your organization transitions to using Tableau Server, an important conceptual shift for you and your users is how you think about sharing workbooks ... and the data that workbooks help you analyze.

Think about data in Tableau Server terms

If you use only Tableau Desktop, you manage all your own connections to data. For example, you might open Tableau, connect to SQL Server, and then select the database, table, and columns to work with. Or you might connect to an Excel spreadsheet and select the sheet to analyze. You generally don't think about how you're going to share the data with others. In fact, a newbie error we've all made is to create a workbook based on a local Excel file, save the workbook as a `.twb` file, and then email the workbook out for others to admire. Only, of course, they can't see anything, because the workbook you've emailed can't actually get to your local Excel file.

Now that you've installed Tableau Server, you and your users must learn a different way of thinking about workbooks and data. Naturally, users will share their workbooks by publishing them to Tableau Server. But you and they also have to think about how to share the data that's used by those workbooks.

With Tableau Server available to your users, you can share data in several ways:

- Create and publish packaged workbooks that contain extracts created in Tableau Desktop. Other users can work with those workbooks and with the static data in the extracts.
- Publish a data source that defines a connection to a database and that includes information about what data in that database to use. Users can then create workbooks that point to this source for their data.

Think about optimizing data access and security

In addition to thinking about ways you can share data access, you and your users must learn how to make the most efficient use of data. Optimizing data access with Tableau Server can seem complex at first. Tableau supports many data connectors. Each connector is optimized for the data that it connects to, and each connector has different characteristics. Many have different authentication requirements. Some do not allow extracts. Some support rich query filtering and operations, while others are more limited.

As you become familiar with Tableau Server and learn how to optimize data access for your scenarios, your users will see these benefits:

- Performance. The goal is "flow." When users are in the flow of data analysis, working with the data in different ways helps them achieve deeper understanding. You want to configure data access so that as much as possible it doesn't interrupt your users' flow experience.
- Access to data. For many organizations, enforcing appropriate access to data is critical to the business. As a Tableau Server administrator, you can make sure that the access that users have to data meets the authentication and privacy requirements of your organization.
- Single source for data. You can use Tableau Server to improve the consistency of data across your organization. You can manage data source connections and create extract refresh schedules to meet the needs of your users and establish stable and consistent data usage.

Before you begin

We've written this chapter for Tableau Desktop champions who have been tasked with managing Tableau Server. Therefore, we assume you understand the differences between a live connection to data (such as a SQL Server or a cloud solution like Amazon Redshift) and an extract. You should be comfortable with the following terms and concepts:

- Data source. A connection to a database or other place where data is stored, with information about what data in that database to use. Users can create workbooks that point to a data source. A data source that is shared on Tableau Server might contain an extract, or it might contain configuration information that describes how to access a live connection.
- Extract. This is a snapshot of data. An extract (`.tde` or `.hyper` file) might be created from a static source of data, like an Excel spreadsheet. Or the extract might contain data from a relational database or from cloud-based data. Extracts that are shared on Tableau Server can be configured to be refreshed from the underlying data according to a schedule that you define.
- Live connection. This refers to a data source that contains direct connection to underlying data, which provides real-time or near real-time data. With a live connection, Tableau makes queries directly against the database or other source, and returns the results of the query for use in a workbook. Users can create live connections and then share them on Tableau Server so that other Tableau users can use the same data using the same connection and filtering settings. As the Tableau Server administrator, you can manage credentials and the permissions associated with the data source to control what data users can access.
- Run As User. This is the Windows account that Tableau Server uses to access data in your organization. We discussed this user account in the planning and installation chapters. The Run As User account can act as the Windows account that Tableau Server uses to get data. It's important if your users need to access data that comes from shared files (such as shared Excel spreadsheets), Microsoft SQL Server, Oracle databases, or other data sources that use Windows authentication.

If you're not already familiar with these different data access terms, take some time to read through Publish Data Sources and Workbooks[(Link opens in a new window)] in the Tableau Help. To refresh your understanding of how Tableau Server uses the Run As User account to access data, review Planning Your Deployment earlier in this guide.

What do you need to do?

This chapter of *[Tableau Server: Everybody's Install Guide]* discusses the three basic steps that you must perform in order to get started in your new role as a data administrator on Tableau Server:

1. Provide access to data sources.
2. Deepen your understanding of the tradeoffs between using a live connection and an extract.
3. Test the performance differences that might occur between a live connection and an extract.

Provide access to data sources

Providing access to data sources starts with understanding how various data sources handle authentication---that is, sign in. In most cases, databases, cloud data, and cubes require users to authenticate before they can access data. The details for authentication are unique to each connector, and authentication is handled by each connector.

As an administrator, you might need to coordinate access to data with the database administrators or data team in your organization. If you *are* the data team, you'll need to understand the data that your organization uses and the authentication requirements that they enforce. For example, when a Tableau user connects to MySQL, Windows authentication is required for access. Users on Tableau Desktop for Windows aren't prompted. But if a user has a Mac, the Tableau Desktop connector for MySQL prompts Mac users for credentials when they attempt to connect.

When possible, we recommend per-user authentication for access to data. However, in some cases the Run As User account can be used to access databases or file shares. One example is SQL Server. If the user publishes a workbook that uses a live connection to SQL Server (not an extract connection), Tableau Server automatically uses the Run As User account when other users access the workbook. This means that when another user opens the workbook, that user's access to data is defined by how much access you as administrator have provided to the Run As User account.

As discussed in the Planning Your Deployment chapter, we recommend that you configure the Run As User for "least privilege" access. In most cases, this means read-only access to the data sources that the account will access.

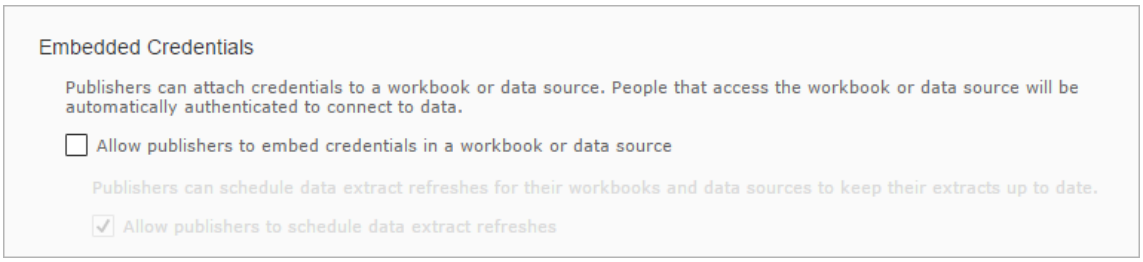
Let users embed credentials

As administrator, you can decide to let users embed passwords (which are encrypted) in the workbooks and data sources that they publish to the server. In that case, when others use those workbooks or data sources, they can see the data without having to provide credentials.

To enable or disable embedded credentials, sign in to Tableau Server.

In the site menu, click [Manage All Sites], and then click [Settings], and then click the [General] tab.

Select or clear the [Allow publishers to embed credentials in a workbook or data source] option.



This is a server-wide settings---the setting you make here applies to all workbooks on all sites.

We should note that there are scenarios where embedded credentials can inadvertently provide access for users who shouldn't see the data. Therefore, for organizations that require user-level authentication to databases, we recommend disabling embedded credentials. Users are then prompted for credentials when they open a workbook, data connection, view, or dashboard that gets its data from a source that requires authentication.

On the other hand, some organizations use Tableau Server as a single managed entry point for data analysts. In this scenario, embedding credentials can make business sense---all the Tableau users who need access to the database can use the single set of credentials that is embedded in workbooks or in data sources.

Set data source permissions

Publishing data sources to Tableau Server lets people on your team provide centralized access to data. It enables data sharing among users, including those who don't use Tableau Desktop but have permission to edit workbooks in the web editing environment. Users working with Tableau Desktop can publish data sources that contain extract or live connections.

As the administrator, you determine which users have the right to publish data sources. These users must have a site role of at least [Publisher] for the site. In addition, non-administrator users must have [View] and [Save] permissions for the project that they want to publish to. (For a review of user permissions, see the Structure Content Projects, Groups, and Permissions chapter.)

Aside from determining who can publish data sources, you can set permissions to determine who can connect to data sources and who can edit them. You can configure access by setting the following permission roles for the project or for individual data sources:

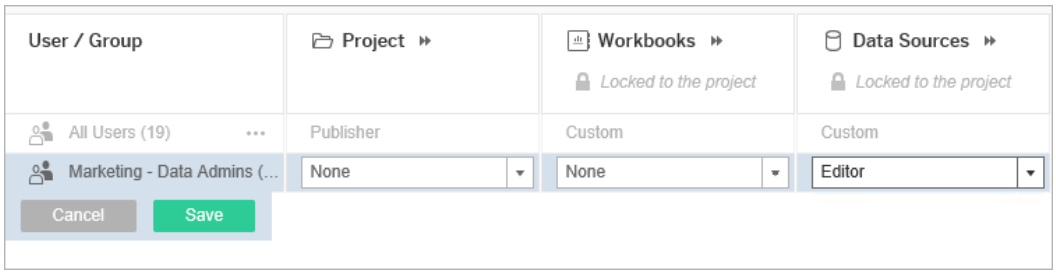
- [Connector]. This permissions role sets permissions that allow the user or group to connect to the data source from a workbook on the server (web authoring) or in Tableau Desktop.
- [Editor]. This permissions role sets permissions that allow the user or group to connect to the data source on the server and also to publish, edit, download, delete, set permissions, and schedule refreshes for the data source.

You can set permissions on individual data sources in a project only if that project is unlocked. As you read in the Structure Content Projects, Groups, and Permissions chapter, we recommend setting permissions at the project level and locking projects after you have configured permissions.

To set permissions on data sources in a project, follow these steps:

1. Sign in to Tableau Server.
2. Click site menu at the top of the page, and then select the site to work with.
3. Click the [Content] tab, click [Projects], and then select the project on which you want to set permissions.
4. Under [Actions], click [Permissions].
5. Select or add the user or group that you want to assign data source permissions to.

   Under [Data Sources], select the permissions role that you want to set.



6. Click [Save].

Share CSV, Excel, or Access files via a live connection

Some of your users might analyze data that's in CSV (comma-separated values), Microsoft Excel, or Microsoft Access files that live as standalone files in a folder. Often these files are treated like a database---for example, several users might be using Tableau to analyze data in an Excel file that's on a shared network location, and someone (perhaps those same users) is also updating the file frequently.

(To be clear, accessing a standalone file on a shared network location is not on a par with using a dedicated, multi-user database like SQL Server, MySQL, or Oracle. CSV, Excel, and Access files don't offer the type of performance, user-level security, or rich querying capabilities inherent in relational databases.)

The procedure we provide here shows you how to share Excel files using a shared folder on a Windows computer that's running in an Active Directory environment.

In this scenario, the Run As User account that you created and configured for Tableau Server is used as the security context to access the Excel file on a network location.

Set up the shared network location

Go to the Share files with someone[(Link opens in a new window)] page on the Microsoft Windows site and follow the procedure under "To share files and folders on a workgroup or a domain." Those steps describe how to use the Windows File Sharing wizard to create a shared folder that is accessible inside your organization using a UNC (universal naming convention). The UNC name consists of a server name followed by a folder name, much like a web address, to access your shared folder. Here's an example, where `DATATEAM` is the name of the computer and `shared` is the name of the shared location on that computer:

```
\\DATATEAM\shared
```

The location referred to by `shared` can actually be many levels deep in the folder hierarchy, even though that's not reflected directly in the UNC name.

When you run the Windows File Sharing wizard, you enter the user account or accounts for people who want to share the content. For this procedure, enter the Run As User account, and then set the [Permission Level] to [Read]. This means that any process that runs as the Run As User can read the shared location. In our case, of course, that process is Tableau Server.

The last page of the File Sharing wizard displays the UNC path. Make a note of this path and send it to your users so they know where to connect to the shared Excel files.

Add the Excel file to the shared location

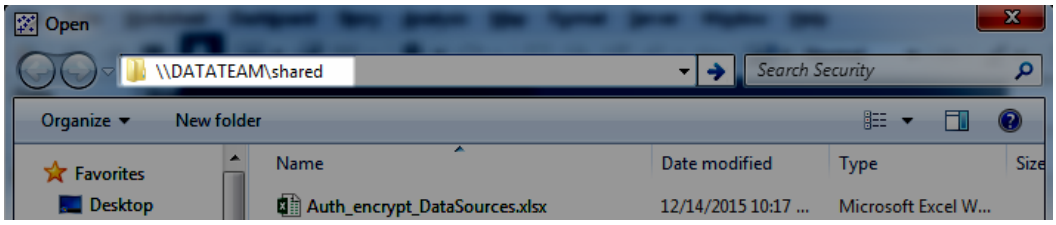
After you've set up the shared location, copy the Excel file (or files) that you want to share to that location.

Create a workbook that connects to the shared Excel file

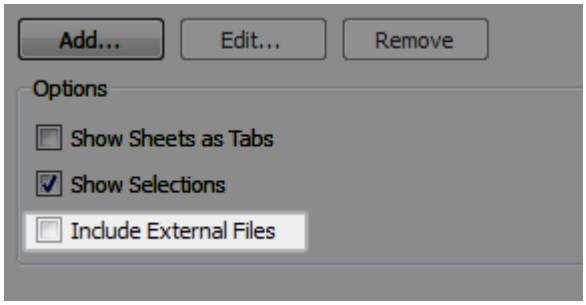
This procedure describes how to access an Excel file on the shared network location and then how to publish that data in a workbook to Tableau Server. This is really a procedure for your users, so make sure they know how to follow these steps.

After users publish using this method, other users who access the published workbook will see data that's coming directly from the shared Excel file. Users can also refresh the data from the Excel file while working in the Tableau workbook.

1. Open Tableau Desktop. On the start page, under [Connect], click [Excel].
2. In the [Open] dialog box, enter the UNC path in the file field at the top, using the format `\\computer-name\share-location-name`.



3. Select the Excel workbook you want to connect to, and then click [Open].
4. Create a Tableau workbook as you normally would, but do not extract the data from Excel.
5. To publish, click [Server] > [Publish]. When you're prompted, enter the Tableau Server address, and the credentials for a Tableau user that has permissions to publish.
6. On the [Publish Workbook to Tableau Server] page, clear the [Include External Files] check box. Click [OK].



7. Set permissions if required, and then click [Publish].

Keep data fresh

In many cases, the data that's displayed in a workbook or view changes after the user publishes the workbook. For example, if a user has a workbook that displays monthly sales information, the data for the workbook has to be updated at least every month.

If the data source for the workbook has been configured to use a live connection to the data, the workbook can read updated data every time the workbook is opened. (That's what we showed you in the procedure just before this section.) But if the data source for the workbook relies on an extract connection, the extract has to be refreshed with the latest data.

When a user publishes a workbook with an extract, the extract is stored on the server. The data is then included with the workbook when a user downloads the workbook or views the workbook on the server.

You can refresh an extract in two ways. A *full refresh* replaces the current extract with new data. An *incremental refresh* adds any new data to the existing extract. (In order to support incremental refreshes, the data has to include data like a date stamp or sequential ID that can be used to indicate where to start the incremental refresh.)

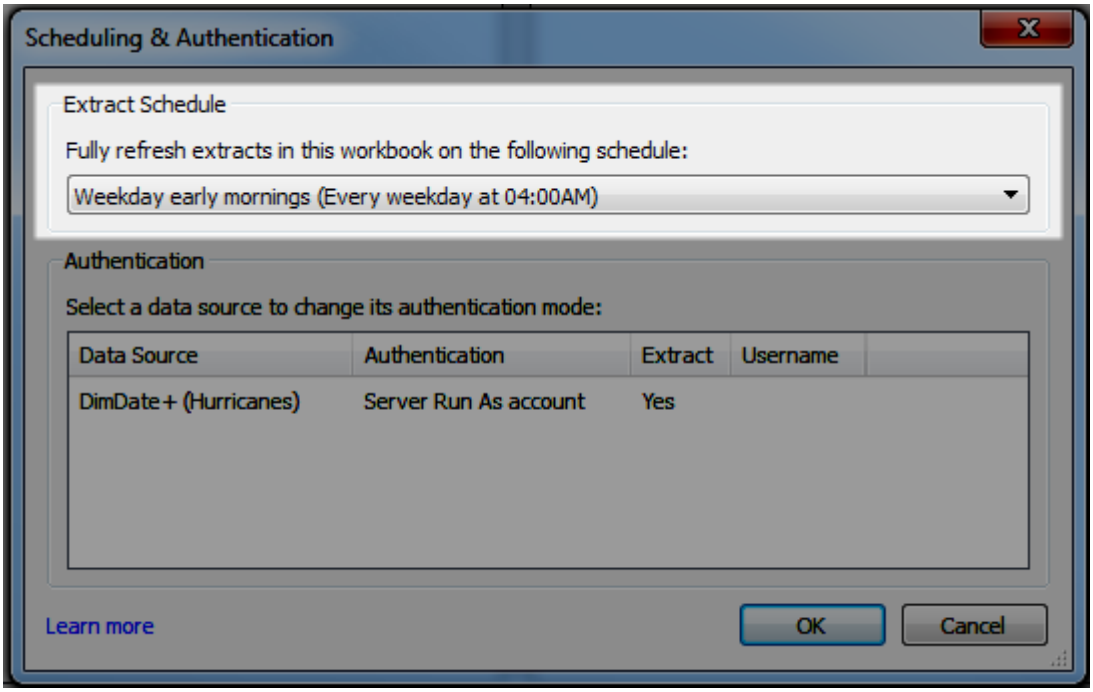
Your users can refresh an extract immediately in Tableau Desktop by selecting a data source on the **Data** menu and then selecting **Extract** > **Refresh**.

As an administrator you can also refresh extracts immediately:

1. Sign in to Tableau Server.
2. At the top of the page, click [Tasks].
3. Under [Extract Refreshes], select the workbook or data source that you want to refresh.
4. On the [Actions] menu, click [Run Now].

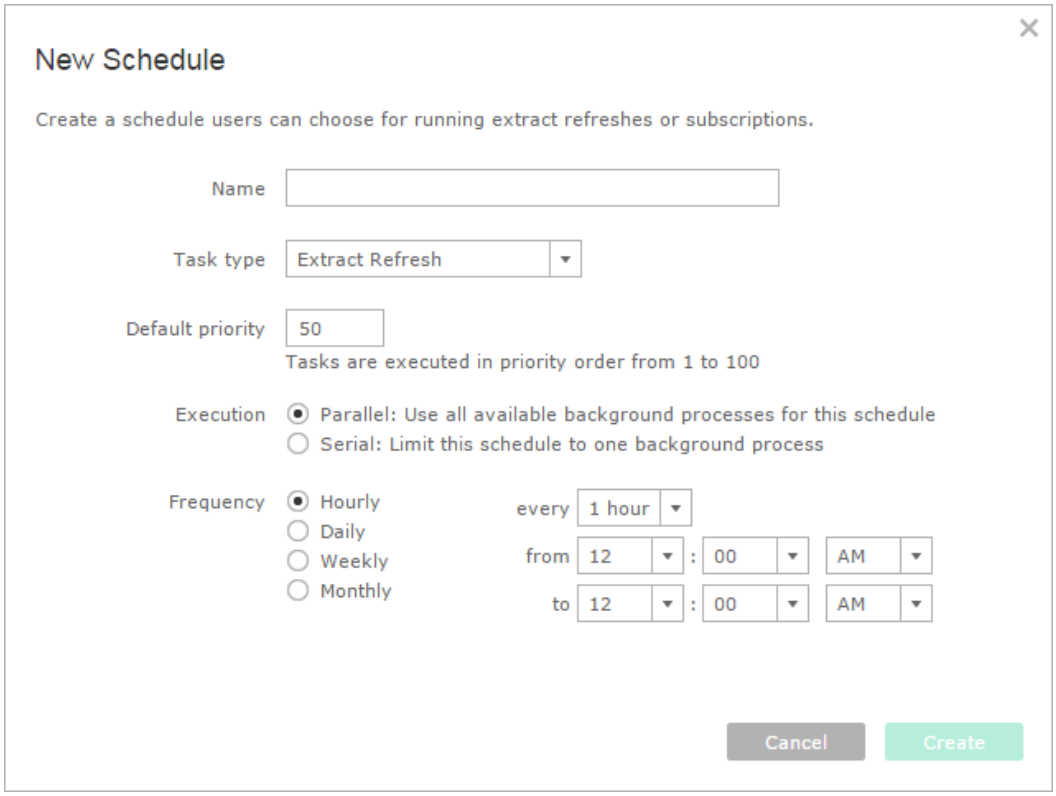
Set refresh schedules that users can choose from

When a user publishes a workbook that includes an extract, the user can set up a refresh schedule. This setting is available from the [Publish Workbook] dialog box in Tableau Desktop.



As the admin, you control the refresh schedule that is presented to users when they publish (the highlighted area in the screen shot). To change the refresh schedules that your users can select, follow these steps:

1. Sign in to Tableau Server.
2. At the top of the page, click [Schedules].
    - Disable, delete, or edit any existing schedules that you don't want by selecting the schedule, and then clicking the appropriate action on the [Actions] menu.
    - Create a new schedule by clicking [New Schedule] and then setting a schedule in the [New Schedule] dialog box:



Determine the frequency of scheduled refreshes

Refreshing an extract can be resource heavy, especially if you're trying to run multiple extract refreshes at once. Therefore, it's generally a good idea to run extract refreshes during non-business hours, and to schedule them as far apart as your business needs allow. A common approach for large extracts is to run incremental refreshes every night during non-business hours, and then run a full refresh over the weekend.

Configure data connection caching

As you plan your data source strategy, you should know how Tableau Server caches workbook data that's hosted on the server. Understanding data connection caching is especially important for organizations that rely on real-time or near real-time data analysis.

Consider a workbook that has a live connection to a database. As users interact with this workbook in a web browser, Tableau Server stores the data that's returned by queries in a cache. That way, if user interaction in the workbook results in a query that's already been issued, Tableau can try to read the data from the cache. Getting data from the cache is usually faster than rerunning a query, which helps the user stay in the flow of their data analysis.

By default, Tableau Server will cache and reuse data for as long as possible. To configure caching behavior for all data connections:

1. Run the following command to set the cache:

   `tsm data-access caching set -r <value>`

   Where `<value>` is one of these options:

   - `low` or empty string (`""`). This is the default value. Tableau Server will cache and reuse data for as long as possible.

- `<n>`. Specifies the maximum number of minutes data should be cached. For example, `tsm data-access caching set -r 2` sets the maximum number of minutes to 2.
- `always` or `0` (zero). Either of these values configure the cache to be refreshed each time a page is reloaded.

2. Apply the changes. Run the following command:

   `tsm pending-changes apply`

In all cases, regardless of how caching is configured, users in Tableau Desktop can click **Refresh Data** on the toolbar to force the server to send a new query and retrieve the latest data. Additionally, users accessing data through a web browser can append the `:refresh` parameter to their URL.

Understand the tradeoffs between using a live connection and an extract

The purpose of this chapter is to guide you through a connection and data source management strategy. A strategy like this tries to answer a fairly simple question: for a given scenario, should your users access live data or should they use extracts?

First of all, some data sources will not allow extracts---they will only allow live connections. Obviously, if that's the case, you don't have to make a decision: use the live connection.

That easy choice aside, there's a long answer to this simple question. As you experiment with different approaches and learn more about the many variables that have an impact data performance, access, freshness, and the ins and outs of specific databases, you'll formulate an answer that works for your users in your organization.

However, we understand that you need to get your users connected to data today. So in this section we provide some guiding principles that you can use to make sound data access decisions as you roll out your shiny new Tableau Server.

Guiding principle: If performance is more important than data freshness, use an extract

Extracts are great for enabling flow for your data analysts. When an extract is embedded in a workbook, all of the data is already available to Tableau Server, which stores the extract in a high-performance database. This generally results in good performance. When users drag dimensions and measures, apply filters, and add visualizations, they see the results immediately. Because users are interacting with a snapshot of data and are not working directly with live data, the underlying source of data is not taxed as users analyze and visualize the data in Tableau.

A note about using Tableau Server for warehousing: if the workbooks that people in your organization are using are really hitting a database hard with repeated queries for fresh data, you might be tempted to use Tableau Server to host extracts in an attempt to offload queries from the relational databases that people are using. Generally, we don't recommend using extracts just to offload queries. This isn't an economical use of Tableau Server, which is designed for data analysis, not data warehousing. If you find that users are creating a lot of extracts because performance suffers when they use live connections to data, you should consider performance optimizations at the database rather than warehousing extracts on Tableau Server.

Guiding principle: If real-time data is required for business decisions, use a live connection

Many data analysis scenarios require real-time data. For example, finance operations that model transactions during trading hours usually require real-time data. Similarly, polling scenarios often require near real-time data freshness to provide quick analysis. Generally, if the data analyses that your users are working on require data freshness that is measured in minutes or seconds, workbooks should be built using a live connection.

Extracts can be refreshed frequently, but as we explained earlier, these updates can be processor-intensive and can slow the performance of the server. At the same time, heavy use of live connections, especially with complex workbooks, can stress traditional databases. Therefore, you'll need to make sure that the Tableau Server processes are appropriately scaled for heavy use of live connections and your databases are up to the task of the query load from Tableau Server.

Guiding principle: If a workbook contains sensitive data, use a live connection

As we were saying earlier, you must decide whether you'll allow users to embed credentials in workbooks and data sources when they publish. Your organization's security and privacy policy should dictate whether you allow users to embed credentials.

If your organization enforces user-level permissions to databases, use a live connection for workbooks that connect to those databases. That way, users who interact with workbooks and data sources that require authentication will be prompted for credentials. For data sources that allow Run As User access, such as SQL Server, Microsoft Analysis Services, and Oracle, make sure that you've configured the Run As User account with appropriate access to the database resources.

Compare the performance of extracts and live connections

People often ask which is faster: an extract or a live connection? If you've read all the way through this, you understand that the answer is "it depends."

In the end, the best way to answer this question is to build a workbook with a live connection to your database. In most cases, the performance differences are obvious as you build your workbook and view the results.

For more in-depth analysis, Tableau includes tools (more information below) that you can use to measure workbook performance on both Tableau Server and Tableau Desktop. Use those tools to profile the performance of the workbook that uses the live connection. When you've got that data, change the workbook to use an extract and then measure performance again.

When you compare these results, a clear winner may emerge. If it's close, you can use the data to guide you in possible ways to improve performance. For example, filtering to use only the subset of data that is required by the workbook might give you an obvious winner.