alteryx

Alteryx Technical Overview

v 1.2, March 2015

Contents

System Overview 3
Alteryx Designer 3
Alteryx Engine 3
Alteryx Service 5
- Alteryx Scheduler 5
- Alteryx Service Controller 5
- Alteryx Service Worker 5
- Alteryx Service Persistence 6
- Alteryx Service Client API and Direct Connect 6

Alteryx Gallery 7
- Alteryx Analytics Gallery 7
- Alteryx Private Gallery 7

Security 8
Deployment 9
- Deploying Alteryx Designer 9
- Deploying Alteryx Server 10

Scaling Out Alteryx 11
- Handling Increases in Workflow Execution Requests 11
- Allowing More Users to Create Workflows 11
- Handling Increased Load on the Gallery 12
- Data Backup and Reliability 13
- Performance and Scaling Considerations 13

Data Connectivity and Sources 14
Appendix 15
Alteryx SDK 15
Alteryx Engine API 15
Alteryx Gallery REST API 16
Integration with Other Systems (Email and Run Command) 17
- Email 17
- Run Command 17

System Overview

The Alteryx Analytics platform offers two deployment options: Alteryx Designer and Alteryx Server.

Alteryx Designer provides business analysts the ability to create repeatable workflow processes for accessing and blending data and performing advanced analytics (such as predictive, spatial, and statistical). Alteryx Designer includes Designer and Engine components and a local version of the Service.

Alteryx Server provides a comprehensive and scalable server-based analytics solution that provides end users and business decision makers the ability to share and run analytic applications in a web-based environment. It also gives users of Alteryx Designer the ability to schedule their workflows to run at specific times. In addition to Designer and Engine components, the Alteryx Server includes Service and Gallery components.

Alteryx Designer

Alteryx Designer is a Windows software application that provides an intuitive drag-and-drop user interface for users to create repeatable workflow processes. Users can drag tools from a toolbox onto a canvas, connect them together, and edit their properties to create Alteryx workflows, apps, and macros. Users can use these workflows to blend and enrich data from a range of sources, perform advanced analytics, and quickly produce results that can be easily shared with others.

The unique drag-and-drop workspace within Alteryx allows users to run workflows as they are being developed in order to instantly access and process data. This allows for continuous enhancements and refinements to the workflow and data results. Additionally, users can view data anywhere within a created workflow stream as a table, map, graph, or other visual representation.

Alteryx Designer is written mostly in C# and executes the workflows through a local instance of the Alteryx Engine. In an Alteryx Server deployment, the Scheduler interface component within Alteryx Designer allows users to schedule workflows to be executed at predetermined times or specific recurring intervals. The Scheduler then communicates with the Alteryx Service where jobs are queued for execution at the appropriate time. Additionally, users may use Alteryx Designer to publish their workflows, apps, and macros to the Alteryx Analytics Gallery where others can have access to running or downloading them.

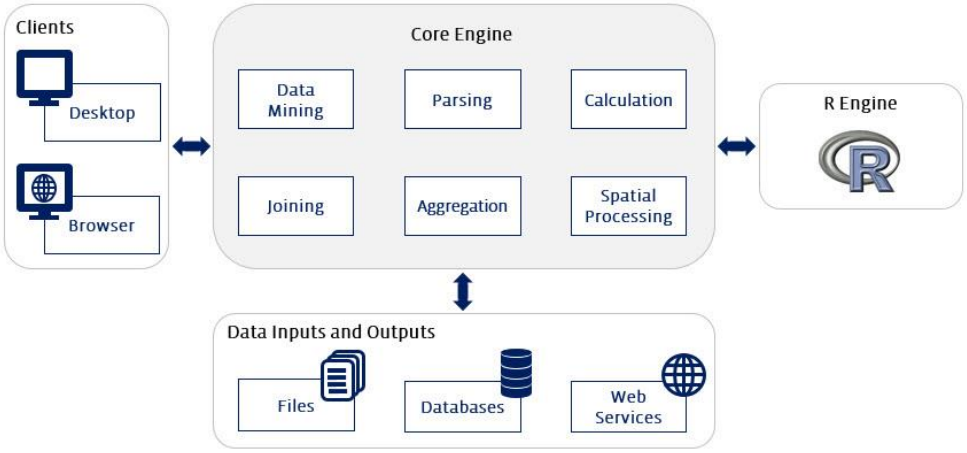
Alteryx Engine

The Alteryx Engine, which is written in C++, executes the workflows that are built in the Alteryx Designer and produces the output. The Engine supports direct connections to various data sources for accessing the data and then processes it in-memory during the execution of the workflow. Processing that exceeds memory limitations is written to temp files on disk, which are deleted once the processing is complete. The Engine can be

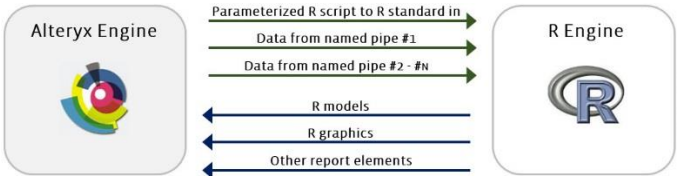
entirely self-contained in an Alteryx Designer deployment, scaled across an organization via the Alteryx Server, or deployed in the cloud via the Alteryx Analytics Gallery.

Depending on the workflow and process, the Alteryx Engine may:

- Read or write to input/output files.
- Read or write to one or more databases.
- Upload or download data from the web.
- Submit email to an email server through SMTP protocols.
- Execute external runtime commands.



The Alteryx Engine seamlessly integrates with R. When users choose to install the suite of R-based tools and macros used for predictive analysis, Alteryx also installs the R program and the additional packages that provide connectivity between Alteryx and R and are required for the tools and macros to work. Users can use R scripts or the R-based macros to process data directly within a workflow. When the workflow is run, the Alteryx Engine communicates with the R engine via a command line executable to process and send back the data.



Developers can write applications that call directly into the Alteryx Engine using the Application Programming Interfaces (APIs) and the Alteryx Software Development Kit (SDK) gives developers an easy way to add their own programs to the Alteryx toolbox. Both the Alteryx API and SDK are available with an Alteryx Server deployment. *For more information refer to the Appendix.*

Whether the Alteryx Engine is running on the desktop, through the API, or via the Alteryx Service, workflows may leverage tools that communicate with databases, output files, web services, email servers, or run commands. *For a list of data connectivity options refer to the Alteryx website:* http://www.alteryx.com/products/technical-specifications.

Alteryx Service

The Alteryx Service allows the Alteryx Engine to be deployed across multiple servers, providing a highly scalable architecture for the scheduling, management, and execution of analytic workflows.

The Alteryx Service, which is written in C++ with some C# wrappers, can be deployed across multiple servers by using a Controller-Worker architecture. This means one server acts as the Controller and manages the job queue, and the others act as the Workers and perform the work. The Service relies upon the Service Persistence tier to store information critical to the functioning of the Service, such as Alteryx application files, the job queue, and result data. The Service also serves content and information to the Gallery when it requests it.

Alteryx Scheduler

The Alteryx Scheduler is the component that provides the ability for a user to schedule the execution of workflows developed within the Designer, either on a recurring basis or at a specific time in the future. The Scheduler interface component, which is written in C#, is accessed via the Designer and is used to specify the date, time, and frequency for running a workflow. The Scheduler then communicates with the Alteryx Service Controller, which manages the job queue. In an Alteryx Server deployment, the Scheduler interface can be set up to communicate with the Controller on the same machine, or it can be configured to connect to a remote Controller so that workflows are processed on a different machine.

Alteryx Service Controller

The Alteryx Service Controller acts as an orchestrator of workflow executions and is responsible for the management of the service settings and the delegation of work to the Alteryx Service Workers. The Controller receives jobs from the Scheduler, looks at them within the persistence layer where it maintains all of the queued jobs, and delegate jobs to the Workers. If Alteryx is deployed across multiple servers, only one machine may be enabled as a Controller.

Alteryx Service Worker

The Alteryx Service Worker contains an instance of the Engine and is responsible for executing the workflows. Once the Controller delegates a job to the Worker, the Worker runs it and produces the output. There must be at least one machine enabled as a Worker in order to execute applications through the Service. You may configure the same machine to be both the Controller and a Worker or they can be configured on separate machines.

The actual number of Workers needed will depend upon the required performance for the system. If the Worker is not the same machine as the Controller, then the connection information for the Controller, which includes the Controller's name or IP address and the unique security token for that Controller, must be specified in order for the Worker and Controller to communicate with each other.

Quality of Service (QoS)

The Quality of Service (QoS) setting is used to manage resource allocation in an environment where multiple Workers are deployed by restricting which jobs will be run by each Worker. For normal operation with one machine configured as a worker, the QoS value should be set to 0. The QoS value for a Worker can be set higher for specific types of jobs to allow the Worker resource to be reserved for higher priority requests.

For example, the default QoS settings in the Alteryx Gallery are:

- Normal workflow execution: QoS = 0
- Chained application execution: QoS = 4 (all apps in the chain aside from the last)
- Workflow validation requests: QoS = 6

When a job request is handled by a Worker, it will compare the QoS value of the job to the Minimum QoS Priority for the Worker. Only jobs that have a value greater than or equal to the Minimum QoS Priority for the Worker will be handled by that Worker. For instance, if a Worker has a Minimum QoS of 0 (the default value) that worker will handle any requests if it is available. However, a Worker with a Minimum QoS of 5 will only handle jobs that have a QoS of 5 or higher. This allows resources to be reserved for higher priority requests.

Alteryx Service Persistence

The Alteryx Service includes a persistence layer that it uses to store information critical to the functioning of the service, such as Alteryx application files, the job queue, and result data. The Service supports two different mechanisms for persistence: SQLite and MongoDB. For lightweight and local deployments, SQLite is adequate for most scheduling needs. For heavier usage, or if the Alteryx Gallery is deployed, MongoDB must be used.

If MongoDB is used, you can elect to either use the embedded MongoDB support native to the Alteryx Service or connect the Service to your own implementation of MongoDB. The embedded MongoDB supported is limited to one server and does not support replica sets or automated backups. It is highly recommended that you provide an automated backup system for whatever persistence mechanism you choose.

Alteryx Service Client API and Direct Connect

The Alteryx Service includes a client API through which it communicates with the Alteryx Gallery and the Alteryx Designer Scheduler to do things such as schedule workflows, view schedules, etc. The client API consists of a low-level C++ library (AlteryxService_Client.dll) as well as a higher-level .NET assembly (AlteryxService_Client.Net.dll).

For performance reasons, the client API can communicate directly with the Alteryx Service Persistence tier for the majority of the requests it makes. This “Direct Connect” functionality is enabled by default but can be overridden if necessary. If Direct Connect is turned off, all communications from the client API will be funneled through the Service Controller.

At startup, the client API will communicate with the Service Controller to determine if the “Direct Connect” functionality is enabled and to obtain the MongoDb connection

information if it is. Then, for requests that can be fulfilled directly from the database, the client API will talk directly to the persistence tier.

The Alteryx Service Client API is currently only consumed internally within the product. External use or consumption of the API is not yet available.

Alteryx Gallery

The Alteryx Gallery is a cloud- or self-hosted application for publishing, sharing, and executing workflows. Alteryx offers the Alteryx Analytics Gallery (https://gallery.alteryx.com/) where users can sign up and share workflows, apps, and macros publicly or with selected users. Additionally, an Alteryx Server deployment allows companies the ability to offer a private Gallery to their internal users hosted on their own server infrastructure.

The Gallery user interface, which is written in Javascript, HTML5, and CSS3, communicates with the Gallery server component via HTTP requests using a REST API, which is a JSON transport protocol. The Gallery web server is written in C# and WCF (Windows Communication Foundation) and handles all the backend logic for what is displayed within the Gallery. WCF receives the HTTP requests and sends them to the server to handle the logic. The server then communicates directly with the MongoDB for handling its persistence, including information such as: the users who have registered with the system, the collections that exist, and the workflows that are in those collections.

The Gallery server also communicates with the Alteryx Service for the management and execution of the workflows. When someone requests a workflow to be executed the Gallery server communicates with the service layer to execute that workflow and then retrieve the output.

Alteryx Analytics Gallery

The Alteryx Analytics Gallery (https://gallery.alteryx.com/), where users can sign up and share workflows, is hosted on an Amazon EC2 deployment and consists of several nodes. Web requests are filtered through the elastic load balancer available within Amazon Web Services. The load balancer sends the requests for the web pages to multiple web servers.

The web servers communicate with the Controller when there is work to be processed. If someone wants to run a workflow that has been published to the Gallery, the web server will communicate with the Controller to schedule the job to be run immediately. The Controller will send the work to the Workers, the Workers will process it, and the results will be sent back to the web server for the end user to see.

Alteryx Private Gallery

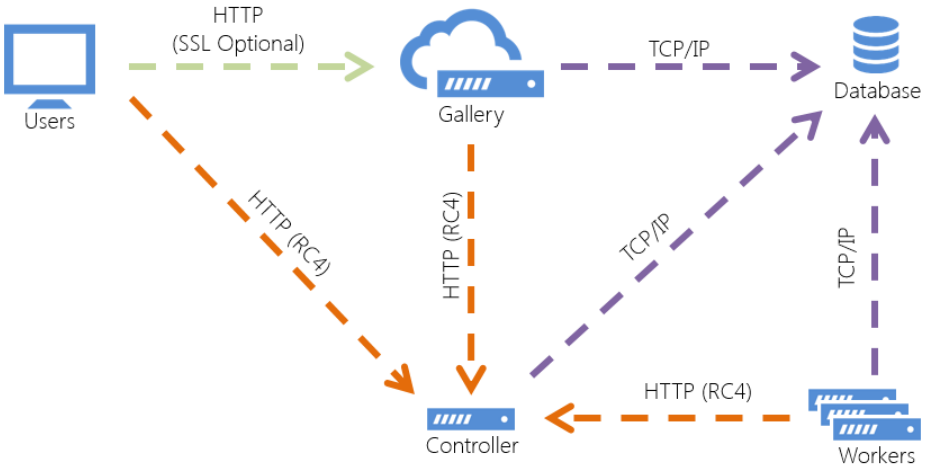
A private Gallery allows organizations to deploy the same architecture used in the Alteryx Analytics Gallery to their internal users within their own IT server infrastructure. This provides a greater level of control over the system as well as allowing internal data to remain inside an organization's firewall.

By default, a private Gallery will be configured to run on a single server, but it can be configured to work across multiple servers in the same way as the Alteryx Analytics Gallery. Each layer of the Gallery architecture is independently scalable, allowing a private Gallery deployment to be customized to each individual organization's needs.

Security

An Alteryx Server deployment that includes a private Gallery uses three main communication protocols:

- Communication between a web browser and the Gallery uses standard HTTP (via port 80). The Server also supports SSL encryption and can be configured to use HTTPs (via port 443). Passwords used when logging in to the Gallery are encrypted using BCrypt and never sent or stored in clear text.
- Communication between a machine that is configured as the Controller and machines that have the Designer, Gallery, or Worker enabled also occurs over HTTP with RC4 encryption that uses the Controller token as the shared secret. The token is auto-generated and encrypted per-machine using the Windows Crypto API.
- Communication between the Service Layer components and the MongoDB persistence layer occurs over TCP/IP (via port 27018). For embedded MongoDB the host, username, and password are automatically generated and the password is encrypted as well. Alteryx also supports SSL for encrypting data when communicating with a user-managed implementation of MongoDB.



Alteryx Server supports two types of authentication to establish a user's identity and prevent unauthorized access to the Gallery: Built-in Authentication, Windows Authentication; and if you are accessing the Gallery via the API you can authenticate with OAuth 1.0A.

- Built-in Authentication requires an email address and password for each user to sign in. User registration information and log-in passwords are encrypted and never sent or stored in clear text

- Integrated Windows Authentication allows users to gain access to the Gallery via their network or Windows user credentials. Instead of storing the password and network credentials of users, Alteryx refers to them via an SID within the Active Directory.
- Alteryx also supports a Gallery REST API which uses OAuth 1.0A via a key and secret for authentication. The key and secret are used only for accessing the API and not for logging in to the Gallery.

With Alteryx Server, authorization for what a user can access and do once authenticated is managed by licensing and user roles and permissions.

- Users must first purchase a license and then activate that license to enable or "unlock" functionality and use Alteryx software, tools, and data. License seats can be shared with other users via a license key.
- Users can be permissioned for roles that govern what they can and cannot do within the Gallery, such as publish, run, and share workflows, or administer the site itself.
- Access to data assets is governed by the database or network permissions that are setup by a company's IT department at the local level drive.

For information on how Alteryx integrates with other systems via Email and Run Command tools and events see the Appendix.

Deployment

The Alteryx Analytics platform offers two deployment options: Alteryx Designer and Alteryx Server. Decisions around selecting a deployment option should take the following into consideration: analytics processing goals, machine specifications, number of users and their needs, automation required to service jobs, access to APIs and SDKs, and volume of data to be processed. *For system requirements refer to the Alteryx website:* http://www.alteryx.com/products/technical-specifications.

Deploying Alteryx Designer

Alteryx Designer provides business analysts the ability to create repeatable workflow processes for accessing and blending data and performing advanced analytics (such as predictive, spatial, and statistical). Alteryx Designer includes Designer and Engine components and a local version of the Service.

A deployment of Alteryx Designer is typically done on a desktop and all data is processed in real time on that machine. Although a Scheduler interface is available via the Designer for scheduling work to be processed at a predetermined time or frequency, it can only be used if there is a connection to a machine that has Alteryx Server installed.

Alteryx Designer may use around 600 MB of space when installed on a single desktop to account for the operating system, application, and data installation. The Engine is designed to make use of the resources available on the machine on which it is deployed and its performance is dependent upon the hardware specifications. The Engine

consumes more memory, processor, and disk space when workflows and apps are executed.

The Engine may run workflows that use tools that communicate with databases, output files, web services, email servers, or run commands. The Engine performance is dependent on the ability to access data and bring it into the workflow, as well as the amount of temp space available for storing files.

Alteryx provides a wide assortment of datasets for analysts who integrate data into their analysis. Depending on an analyst's needs, an install of the data may use up to 180 GB of space on a machine. *For information about available datasets refer to the Alteryx website.*

Deploying Alteryx Server

Alteryx Server provides a comprehensive and scalable server-based analytics solution that provides end users and business decision makers the ability to share and run analytic applications in a web-based environment. It also gives users of Alteryx Designer the ability to schedule their workflows to run at specific times. In addition to Designer and Engine components, the Alteryx Server includes Service and Gallery components.

An Alteryx Server deployment provides for the scheduling and sharing of the workflow processes that have been built via the Designer. Additionally, once the Server is installed it is scalable to accommodate for higher levels of usage by enabling certain components on additional machines.

With the Alteryx Service (which includes the Controller, Worker, and Persistence) jobs can be scheduled to be executed at predetermined times, or at specific recurring frequencies, by connecting to the Controller via the Scheduler interface in the Designer. The Service manages the scheduled job queue, how the job processes, and the results and logs associated with the job in a persistence layer that can be either SQLite or MongoDB. Both SQLite and MongoDB are included in the Server installation, or an existing MongoDB instance may be used.

The Alteryx Gallery component is a web-based application that allows users to publish, share, and execute workflows and apps in a browser. The Gallery web server communicates with the Service for the management and execution of the workflows and a MongoDB persistence layer for maintaining information about the users who have registered with the system, the collections that have been created, and the workflows and apps that are in those collections. When deploying Alteryx Server, companies can take advantage of all the functionality provided in the Alteryx Analytics Gallery while offering a private Gallery to internal users hosted on their own server infrastructure

The Alteryx Service is composed of several different processes that may be running on the machine to handle various aspects of the Alteryx Server functionality such as the Gallery web interface, execution and scheduling of the workflows, and the database persistence layer. To ensure system stability, the Service continually monitors these processes and if a process is not running, the Alteryx Server will try to start it back up. If it is unsuccessful after multiple attempts, the Alteryx Server will stop all of the other processes, shut down, and log the problem.

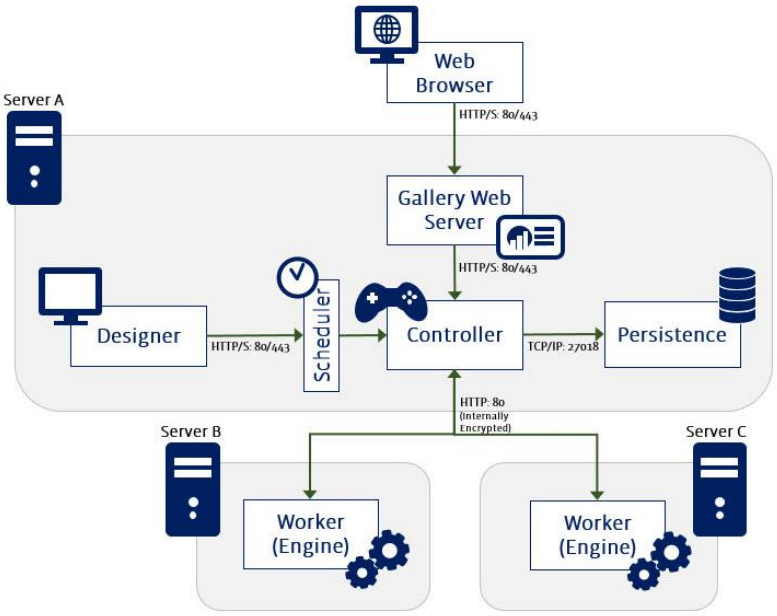
Scaling Out Alteryx

The complete Alteryx Server can be enabled on a single machine, selected components can be enabled on a machine, or certain components can be enabled on multiple machines to accommodate for higher levels of usage.

A typical deployment of Alteryx Server includes installing the Controller, Worker, and Gallery web server component, on a single machine. There can only be one instance of the Controller, but the Worker, Designer, or web server components can be enabled on individual and multiple machines as the need to handle different workloads increases. Decisions around which Alteryx components to scale out are based on the type of work that is needed and who is doing it.

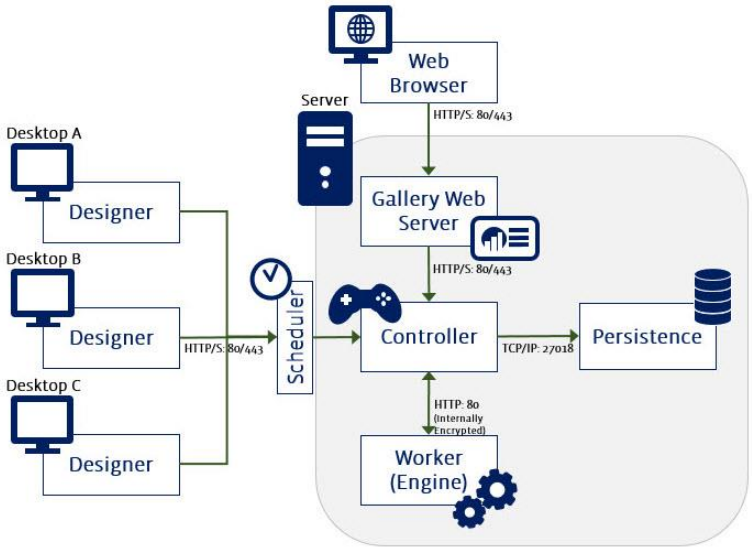
Handling Increases in Workflow Execution Requests

If more work needs to be processed and there is not enough simultaneous processing to be able to handle it, the Engine processing capabilities can be increased by configuring multiple machines to act as Workers. Each additional Worker machine must be configured with the unique security token of the Controller to be able to communicate with it. Then, as workflow execution requests begin to queue up, each Worker machine will communicate with the Controller and the Controller will delegate a job for it to process.



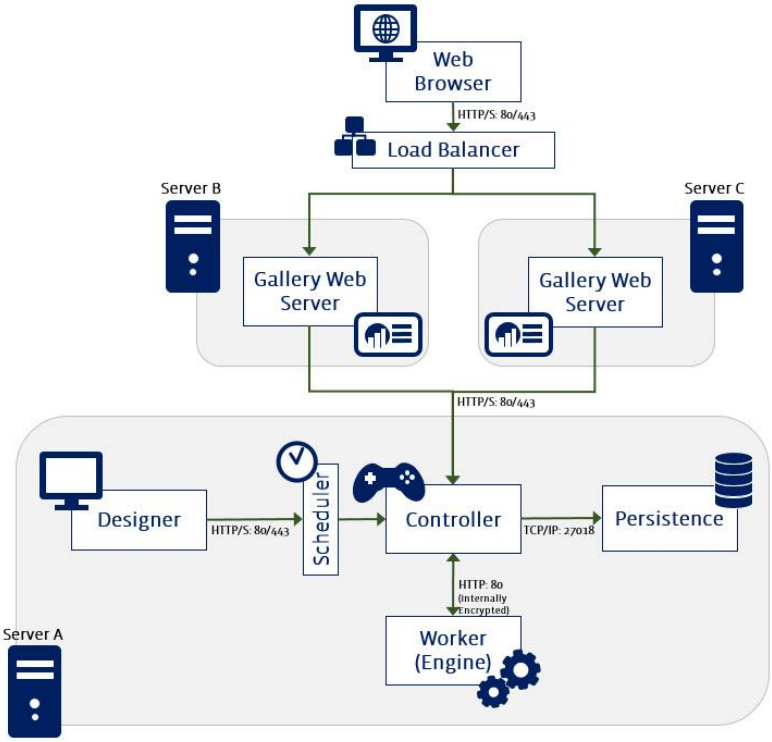
Allowing More Users to Create Workflows

If there are multiple end users using more lightweight machines to workflows in Designer within their own desktop, each machine can be configured to communicate to the same Controller so jobs can be scheduled to be processed on a more robust Worker machine. The Worker will execute the job and provide the output which the users can retrieve from their desktop.



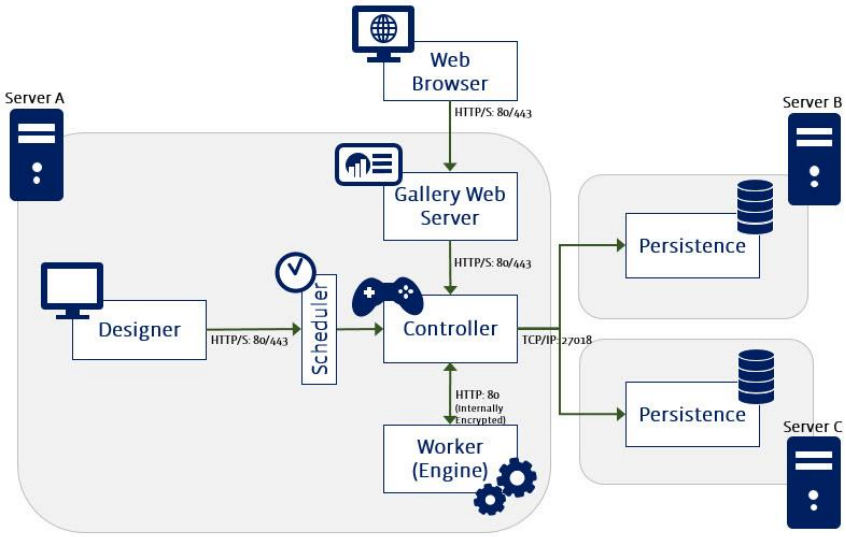
Handling Increased Load on the Gallery

If web requests increase because there are many users who need to access or share the workflows and apps published to the Gallery, the Gallery web server can be scaled out by configuring additional machines to act as web server nodes. When setting up multiple web server nodes these nodes would generally be behind a load balancer to help distribute the amount of web requests across all the web servers. Each individual web server will then communicate to the same Controller when there is work to be processed.



Data Backup and Reliability

If a machine is configured with only the Designer and Scheduler, SQLite can be used as the persistence layer. However, if the Gallery component is enabled on the machine as well, MongoDB must be used. Alteryx includes an embedded version of MongoDB for ease of setup and use but the embedded MongoDB is limited to one server and does not support replica sets or automated backups. If you deploy your own instance of MongoDB and manage it yourself, you can follow the MongoDB recommendations for scalability and redundancy.



Performance and Scaling Considerations

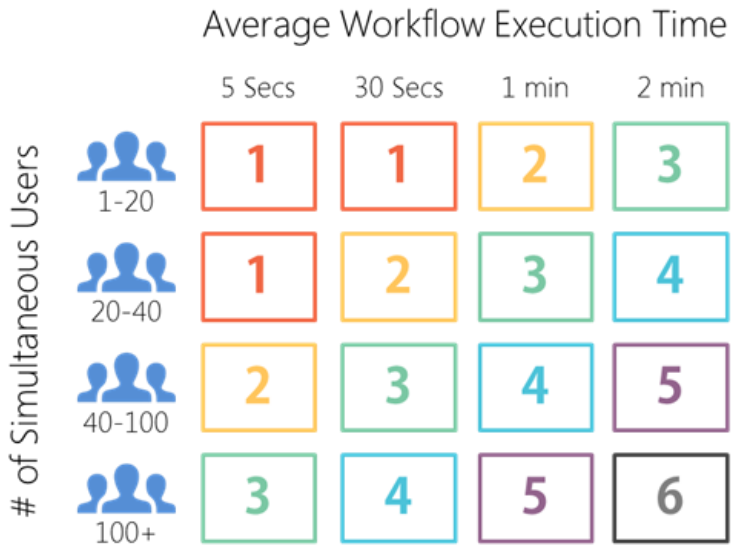
Multiple factors should be considered when making decisions for a Server deployment. Machine specifications, number of users and their needs, types of workflows being built, automation required for jobs, and volume of data to process are all things that will impact the type of hardware and the configuration needed for optimal performance. In general, optimizing hardware improves processing speed and adding more hardware increases processing capabilities for multiple concurrent users.

A typical deployment of Alteryx Server includes installing the Controller, Worker, and Galley web server component, on a single machine. There can only be one instance of the Controller, but the Worker, Designer, or web server components can be enabled on individual and multiple machines as the need to handle different workloads increases.

The following table provides some guidelines for determining when more hardware for Alteryx Server components may need to be added. A load balancer may be needed to increase capacity and distribute web requests when additional Gallery nodes are added.

	Worker	Gallery	Database
Redundancy needed	✓	✓	✓
High availability needed	✓	✓	✓
Workflow execution time increases	✓		
Number of simultaneous users increases	✓	✓	
Number of backend jobs increases	✓		
Workflows sit in queue for long periods of time	✓		
Memory or CPU consumption is high on web nodes		✓	

An Alteryx Server deployment often involves configuring multiple machines to act as Workers. The following image provides guidelines for determining the total number of Workers that may be needed, based on the total number of simultaneous users and the average amount of time it takes for a workflow to execute.



In some cases, performance and processing speed can be improved by optimizing the hardware.

- Using a Solid State Drive over a Hard Disk Drive could improve processing speeds for workflows involving CASS, Calgary, and geocoding functions.
- Increasing the CPU speed may result in faster processing and lower latency.
- Adding more memory, based on the amount of data to be processed, may improve performance.
- Increasing the number of cores could improve processing speeds for workflows involving drive time and CASS.

Data Connectivity and Sources

Alteryx provides data connectivity options to enable users to access and integrate data from many different sources. The various types of data sources Alteryx supports include relational databases, document stores, flat files, and web services.

Through partnerships with leading data providers, Alteryx also provides a wide assortment of datasets for analysts who require spatial or syndicated data to be integrated into their analysis. Some of this data is included when Alteryx Designer or Alteryx Server is installed and additional data can be added after installation.

Data is bundled into packages in a way that allows users to easily install all data at once or to install specific datasets within a package individually. The complete data package may be up to 180 GB in size when installed on a machine or network.

For a current list of data connectivity options and information about available datasets refer to the Alteryx website.

Appendix

Alteryx SDK

The Alteryx SDK allows developers to create their own tools for use within Alteryx workflows. The SDK is available as a set of C++ libraries or through a .NET wrapper. All of the tools in the Desktop Designer were built with the Alteryx SDK.

Alteryx workflows consist of a series of tools connected together. Each of these tools falls into one of three categories:

- Input tools, which generate data either spontaneously (such as a random number generator) or by reading information from a data source (such as a file or database).
- Output tools, which consume data either displaying it (such as a table or map) or storing it to a data source.
- Computation tools, which consume data, process it, and then generate new or additional data.

All tools, regardless of their type, are implemented in a nearly identical fashion. Each tool has a user interface implementation, which handles the visual rendering of the tool object as well as its configuration properties dialog. Each tool also has an engine implementation, which performs the actual processing work. These two pieces together form the single tool.

When needed, Alteryx calls into a tool's user interface implementation to provide its Properties view. When the workflow runs in the Alteryx Engine, an instance of each tool's engine implementation is created, and the outputs of upstream tools are connected to the inputs of later tools. Alteryx then invokes each input tool in the workflow. During the run, tools push their output directly to one another, minimizing execution overhead.

User interface implementations can be written in any .NET language. Alteryx uses C# for this purpose, but others have been successful at using VB.NET. Engine implementations are written in any language which can produce DLLs with an unadorned C-style binding. Alteryx uses Visual C++ for its engine implementations, with a few exported C-style functions to provide the binding layer.

Alteryx Engine API

The Alteryx Engine API provides multi-level integration access to both manage the Alteryx Engine and also work directly with Alteryx supported data formats. The APIs consist of native C++ dlls with.NET wrappers that allow you to access various portions of the Alteryx functionality.

Included with the API are two high-level .NET wrappers for running Alteryx workflows.

- AlteryxAPI.Net.dll: This assembly includes classes for running workflows, accessing properties and settings for Alteryx, as well as high-level classes for accessing Allocate, Solocast and Calgary functionality.
- AlteryxDocument.dll: This assembly provides access to Alteryx workflow properties and settings, including runtime settings for apps and macros.

In order to work with the API, simply add the appropriate assemblies to your .NET project. The Alteryx installation includes documentation with its APIs in order to help get you started. There are also direct APIs for specific functionality (Allocate, Solocast, Calgary, etc.).

Alteryx Gallery REST API

The Gallery REST API provides you with a web service API that you can use to interact with your Private Server Gallery, or the Alteryx Analytics Gallery, when creating your own interfaces for running apps.

The Gallery API gives you the ability to:

- Authenticate
- Retrieve apps in a subscription
- Retrieve app questions
- Run apps
- Retrieve app jobs
- Retrieve app status
- Retrieve app results

Each resource in the API is a named URI, and are accessed via HTTP, or HTTPS if enabled. Data transport to and from the API is via JSON. Authentication is done via OAuth 1.0a using an API Key and Secret which can be obtained from a user's Private Studio Settings.

Interactive API documentation to help developers learn elements of the API, along with example code in various programming languages, is available at http://[your gallery url]/api-docs/ for the Private Server Gallery and at http://alteryx.gallery.com/api-docs/ for the Alteryx Analytics Gallery.

Integration with Other Systems (Email and Run Command)

Alteryx can be expanded further and integrated into other systems by way of Email and Run Command tools and events.

Email

Using the Email tool or event, Alteryx can prepare outgoing emails and submit the emails through the SMTP protocol to any SMTP-enabled email server. The user can:

- Set the To, From, CC, and BCC fields.
- Create the subject line.
- Create the body of the message.
- Attach documents.

Run Command

Using the Run Command tool or event, Alteryx allows for the execution of command line functions through the "Run" shell. Common uses for this command include integration through:

- Batch commands to execute external processes.
- Complex file transfers.
- Custom scripts or other add-ons.

Document History

Version	Date	Description
v 1.0	August 2014	Initial document.
v 1.1	January 2015	Added where to find instructions for enabling SSL and installing a certificate. Updated “module” references to “workflow” throughout.
v 1.2	March 2015	Added additional details in the Security section. Added the Performance and Scaling Considerations section.