



iRODS

Technical *Overview*

The integrated *Rule-Oriented* **DATA** System

iRODS is open-source, data management software that lets users:

- access, manage, and share data across any type or number of storage systems located anywhere, while maintaining redundancy and security, and
- exercise precise control over their data with extensible rules that ensure the data is archived, described, and replicated in accordance with their needs and schedule.

iRODS empowers users by supporting

- virtualization, which provides a one stop shop for all data regardless of the heterogeneity of storage devices,
- data discovery through the use of descriptive metadata,
- workflow automation, and
- data sharing between collaborating or distributed teams.

This document provides a brief overview of the iRODS architecture to help you consider whether iRODS is the right solution for your data management needs. It will describe the ***iRODS Metadata Catalog (iCAT)***, a database that stores all the valuable information about your data, the ***iCAT-Enabled Server (IES)***, and optional ***Resource Servers*** which facilitate data management and replication throughout your storage infrastructure. Then, the document will describe facets of the architecture that may be customized to precisely control and manage data. These facets include policies, rules, plugins, and ***Composable Resources***. This document concludes with some questions that may be helpful for determining if iRODS will serve your needs, followed by pointers to other helpful information about iRODS.

AN iRODS ZONE

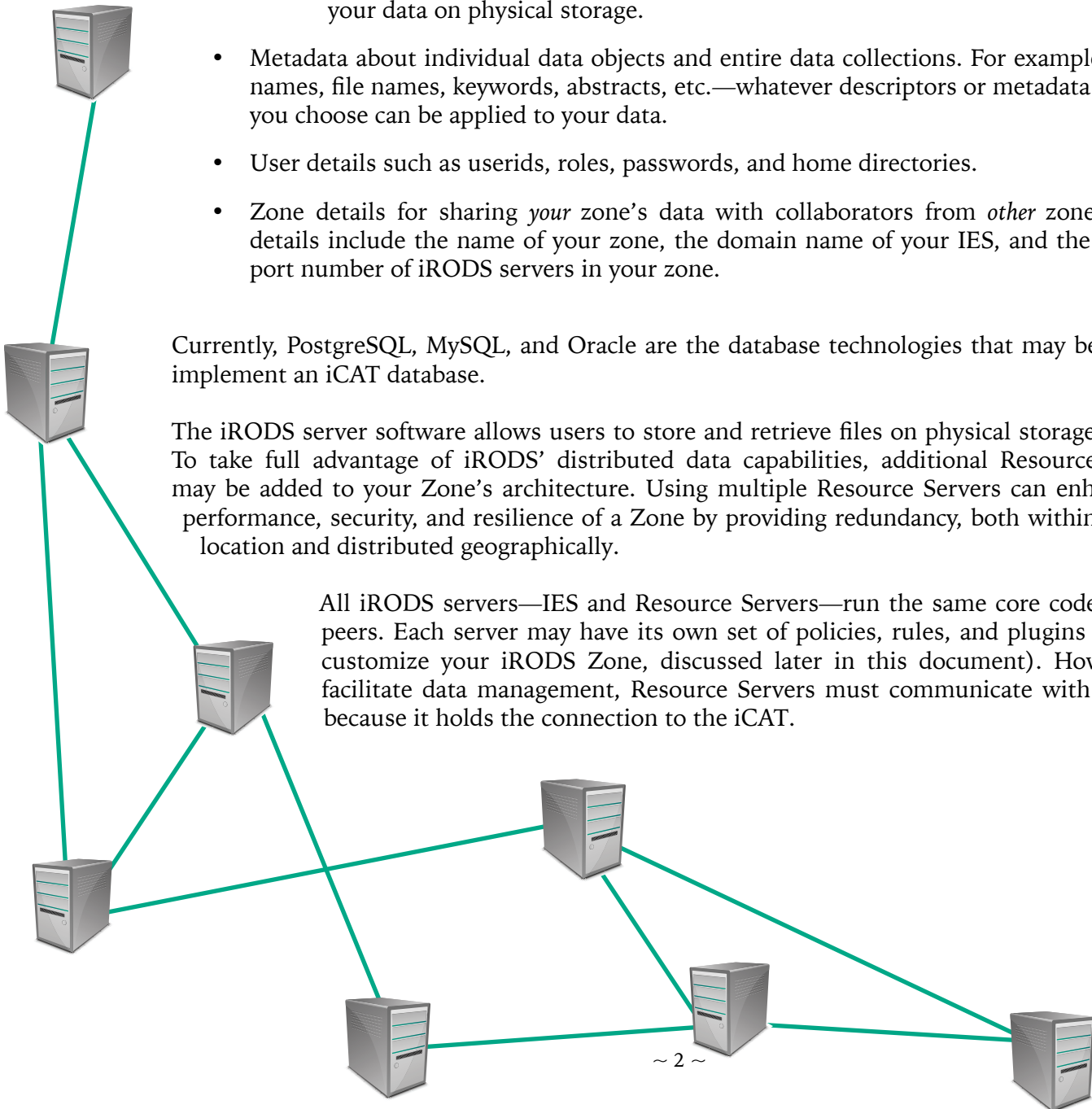
Each iRODS deployment—or **Zone**—is composed of an iRODS Metadata Catalog (iCAT), an iCAT-Enabled Server (IES), and optional Resource Servers. The iCAT is a relational database (i.e., a SQL database) that holds all the information about your data, users, and zone that the iRODS servers—IES and any Resource Servers—need to facilitate the management and sharing of your data. The iCAT contains the following information:

- Data object (i.e., file) location and access information
 - A virtual file system (i.e., mapping of logical directory paths to actual physical storage paths), which provides a unified namespace across various underlying storage systems.
 - Resource configuration information—hostnames and credentials—for accessing your data on physical storage.
- Metadata about individual data objects and entire data collections. For example, author names, file names, keywords, abstracts, etc.—whatever descriptors or metadata schemes you choose can be applied to your data.
- User details such as userids, roles, passwords, and home directories.
- Zone details for sharing *your* zone's data with collaborators from *other* zones. These details include the name of your zone, the domain name of your IES, and the network port number of iRODS servers in your zone.

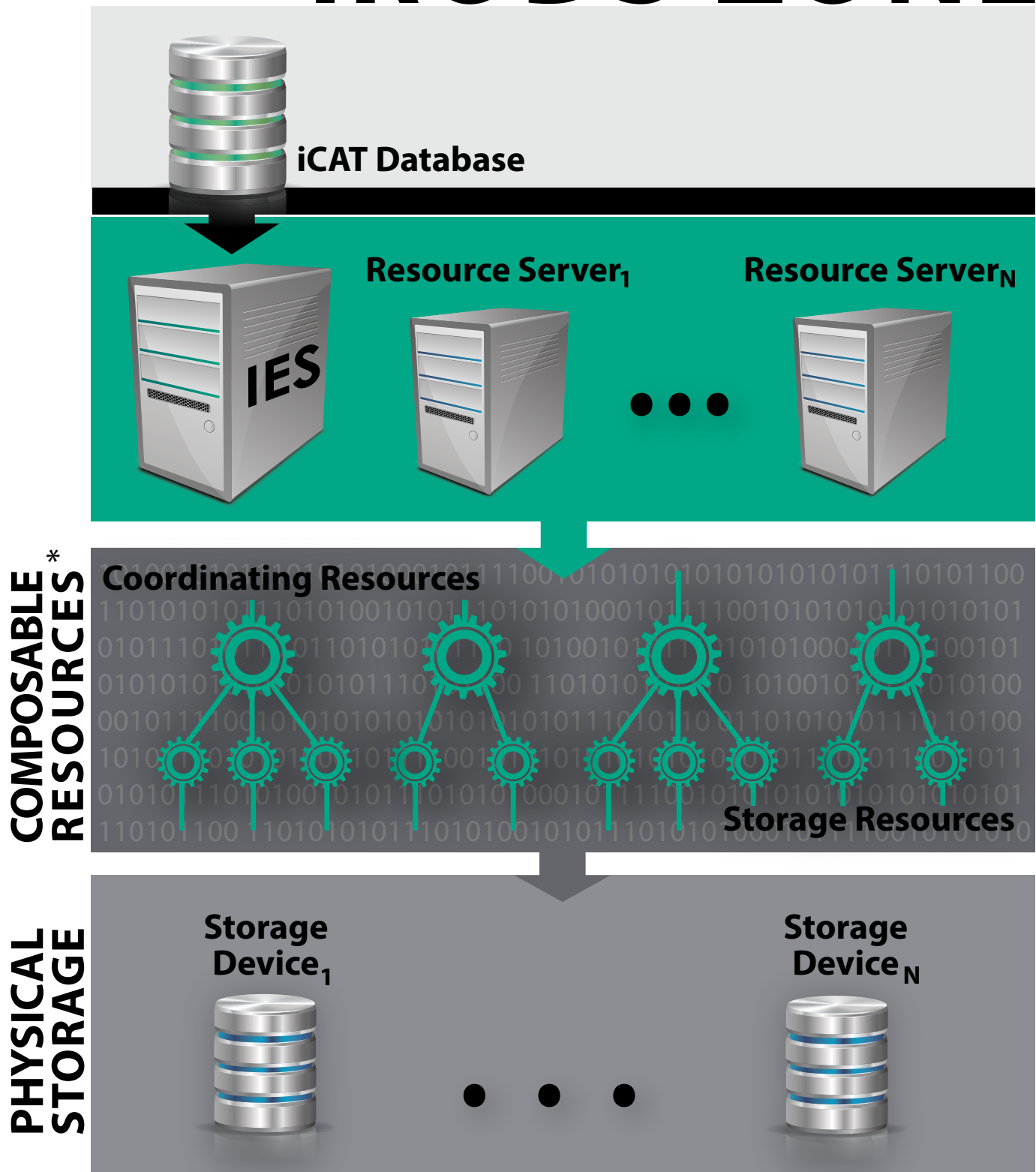
Currently, PostgreSQL, MySQL, and Oracle are the database technologies that may be used to implement an iCAT database.

The iRODS server software allows users to store and retrieve files on physical storage devices. To take full advantage of iRODS' distributed data capabilities, additional Resource Servers may be added to your Zone's architecture. Using multiple Resource Servers can enhance the performance, security, and resilience of a Zone by providing redundancy, both within a single location and distributed geographically.

All iRODS servers—IES and Resource Servers—run the same core code and are peers. Each server may have its own set of policies, rules, and plugins (ways to customize your iRODS Zone, discussed later in this document). However, to facilitate data management, Resource Servers must communicate with the IES, because it holds the connection to the iCAT.



iRODS ZONE



**You may arrange composable resources according to your needs.*

POLICIES

Your organization has data management policies that specify processes for access control, backup, data migration, data preparation, metadata extraction, and more. These organizational policies can be implemented as iRODS policies, which can help you customize and automate these data management tasks. iRODS policies consist of a rule or set of rules that are triggered based on a condition or set of conditions. These conditions may pertain to timing, user role, physical location, etc.

Rules

Rules are written in iRODS' domain-specific rule language, which uses many familiar programming constructs (e.g., loops, conditional statements), making it easy for your organization's developers to construct rules to satisfy your data needs.

Rules are carried out by the iRODS rule engine—a built-in interpreter for the iRODS rule language—that governs the sequence of data management actions in your iRODS zone. The IES and each Resource Server run an instance of the rule engine. Out of the box, the rule engine comes loaded with an array of basic actions (e.g., error reporting, login protocols) to get you up and running. As you begin to dig into iRODS, you can add rules of your own to tailor a data management program that works for you.

Rules usually call one or more *microservices*—pre-packaged C++ programs designed to accomplish specific tasks. Over 350 microservices have been contributed to the iRODS codebase by the user community and development staff. Using iRODS microservices, rather than calling a standalone external program, enables enforcement of iRODS access control policies, code verification for auditing and scientific reproducibility, and coordinated code execution across multiple iRODS servers.

You will notice in the example rule on the next page that one microservice (`msiDataObjRep1`) is called. Microservices are denoted by the prefix *msi*. They are a specific class of plugins, discussed in the Plugins section of this document.

iRODS *rules* ENABLE...

- **Access Control**
Rules can specify who can access data, for what activity (i.e., read or write), and when they can access it.
- **Backup and Data Migration**
Rules can manage file replication and transfer among storage devices or locales, depending on any parameter, such as date of creation or last access.
- **Data Conversion**
Rules can convert files between file formats or different versions of the same format, such as when you need an Excel spreadsheet saved as XML.
- **Data Preparation**
Rules can clean and prep your data prior to use; for example, eliminating duplicate data, removing null values, performing spell checks, indexing data, subsetting data into more manageable chunks, or any pre-processing that meets your needs.
- **Metadata Extraction**
Rules can extract relevant metadata from files to improve search, so you are not stuck remembering file names.

An example rule

```
ReplicateAllFilesWithExtension {
  *Query = SELECT COLL_NAME, DATA_NAME WHERE DATA_NAME LIKE "%. *Extension";
  foreach(*Row in *Query) {
    *SourceFile = *Row.COLL_NAME++/"++*Row.DATA_NAME;
    msiDataObjRepl(*SourceFile,"destRescName=*Resource",*Status);
    writeLine("stdout","Replicated *SourceFile to *Resource");
  }
}
INPUT *Extension="txt", *Resource="backupResc"
OUTPUT ruleExecOut
```

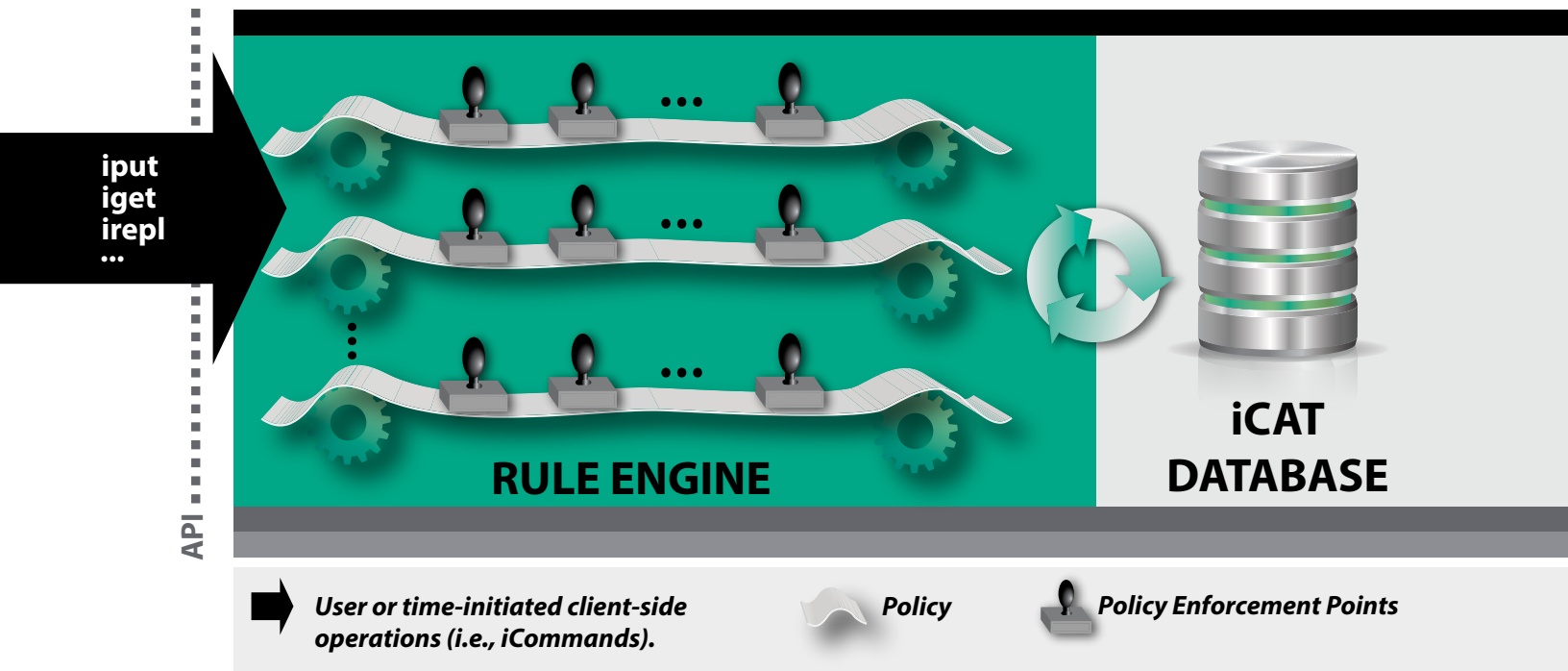
The example rule above locates every file with the .txt extension and creates a replica of each located file on the resource called backupResc.

Policy Enforcement Points (PEPs)

Rules are executed based on conditions or, in iRODS parlance, *Policy Enforcement Points (PEPs)*. Consider, for example, a rule to transfer ownership of data objects to the project manager when a user is deleted; the trigger—or PEP—is the deletion of the user. Similarly, rules could be written to extract metadata or pre-process data whenever a file is uploaded to a storage device. Or, upon access to particular data objects, a rule can create a log of the event, send an email notification to the project manager, or perform some other task you need to occur as a result of the data's access.

As illustrated in the diagram below, every client-side operation (e.g., storing a file or *iput*, retrieving a file or *iget*) activates a PEP which can trigger any action you need iRODS to take to manage your data and metadata. The chaining of rules and PEPs allows you to create powerful, customized workflows that help you automate, save time and energy, and prevent human error.

INSIDE iRODS






PLUGINS

In addition to rules, an iRODS installation can be customized with plugins. Plugins are used to implement core iRODS functions, such as authentication, communication over the Internet, communication with storage devices, and more. The use of plugins allows administrators to tailor iRODS to their needs and to the existing infrastructure, without having to re-compile code. Plugins also make it possible to upgrade small portions of iRODS without interfering with core functions.

Plugins interact with iRODS core code through six interfaces. You have already read about one pluggable interface, microservices. The other five are authentication, communication with the iCAT database, network transport, the application programming interface (API), and communication with storage devices. iRODS ships with a default set of plugins; others must be installed separately, if you choose to use them.



Interface	Plugins
Authentication	Native iRODS password access OSAuth Pluggable Authentication Module (PAM) Grid Security Infrastructure (GSI)* LDAP via PAM
Database	Oracle* PostgreSQL* MySQL*
Network	Transmission Control Protocol (TCP) Secure Socket Layer (SSL)
API	An iRODS application programming interface to enable storage and retrieval of data objects from another application.
Microservices	There are over 350 available, covering a variety of functions.
Composable Resources	There are two kinds of Composable Resources in iRODS: Coordinating and Storage. Both are discussed in more detail in the next section.

**External plugins that may be installed*



COMPOSABLE RESOURCES

Composable Resources are plugins that allow you to create rich decision trees for managing storage and retrieval of data on storage devices. There are two types of Composable Resources: *Coordinating* and *Storage*. Coordinating Resources actively make decisions about which physical device will receive or serve up a data object. Storage Resources are the logical representations of—or pointers to—physical storage devices. They are composed of five parts: (a) the name you give the resource, (b) a host name (e.g., willow.irods.renci.org), (c) a directory path to the exact location on the storage device (e.g., /full/path/to/storage), (d) the resource type (e.g., Amazon S3), and (e) a plugin-specific context string (e.g., the name of the file containing access credentials or any persistent information the plugin may require).

One way to think about Composable Resources is to consider Coordinating Resources as the branch nodes of your decision tree and Storage Resources as the leaves. (See the iRODS Zone figure.) Examples of Coordinating Resources include:

Replication	Round Robin	Load Balanced	Compound
A replication resource keeps its children in sync with identical copies—or replicas—of data objects.	A round robin resource will rotate through its children for each upload to the system.	A load balanced resource attempts to balance storage and retrieval across children to avoid taxing the servers (e.g., available space, CPU usage, network traffic).	A compound resource manages two children, in the roles of <i>Cache</i> and <i>Archive</i> . The Cache resource provides a standard UNIX file system interface (i.e., POSIX) to an Archive that may not natively support this type of access.

Examples of Storage Resources that may be used in an iRODS installation:

Unix File System	This type of storage resource communicates with a storage device through the standard POSIX interface.
Amazon S3*	Amazon's cloud storage service.
Web Object Scalar (WOS)*	DataDirect Networks' block storage appliance.
Ceph RADOS*	Designed for efficient cacheless access to a Ceph RADOS block storage cluster.
Direct Access	Designed for access to shared high performance computing (HPC) storage.
Universal Mass Storage	For use with Compound resources, such as tape-based archives.

**External plugins that may be installed*

IS iRODS RIGHT FOR ME?

iRODS provides a customizable solution for your data management needs. Questions you may want to ask yourself when evaluating iRODS as a potential solution include:

- Do I need to share, back up, or migrate data between different types of storage devices and/or different geographic locations now or in the future?
- Do I need to move data easily between local and cloud-based storage devices, or file and block storage devices?
- Could my collaborators and I benefit from accessing and managing data in a unified namespace?
- Do I need to automate data management activities (e.g., data replication, load-balancing) and/or could I take advantage of data processing workflows to streamline data management?
- Do I need to implement and verify compliance with regulatory data retention and privacy policies (e.g., HIPAA, Sarbanes-Oxley, or other federal regulations)?
- Do I need to provide data description and annotation (e.g., metadata) for improved search, sharing, or data provenance tracking?

iRODS can help you fulfill all of these data management needs. Also consider your future needs. Maybe you don't need to share data with collaborators or outside organizations yet; but is it possible that you might need to scale up at some point to do so? Maybe you will want to migrate storage hardware some day. iRODS can grow with your organization as your data needs grow.

For more information:

- iRODS website: <http://irods.org>
- iRODS Consortium: <http://irods.org/consortium>
- iRODS documentation: <http://irods.org/documentation>
- iRODS code on GitHub: <https://github.com/irods>