Chapter 4

# Deploying and Maintaining a Repository in the Cloud

Cloud infrastructure is, by definition, virtual infrastructure. The server instances, storage blocks, routing instructions, and every other cloud function are facilitated by an underlying code base; while these services are supported by hardware, it is code that defines and enables these resources to function in the same manner as their physical counterparts. And it is because they are created in code that every cloud resource can be accessed, configured, and controlled through a web interface, an API, or a command line tool. These various methods of control allow an institution to use a variety of sophisticated means of managing its infrastructure and the applications (e.g., digital repositories) that it supports.

## Infrastructure as Code

Infrastructure as code (IaC) is a concept that became possible with the advent of hardware virtualization.[1] The concept can be described as using templates, scripts, or some other machine-actionable documents to describe a virtual infrastructure, and in doing so enabling the automatic instantiation of said infrastructure. In other words, defining cloud IaC allows for the automatic creation of cloud resources. The benefits of this practice are numerous; simply having all its IT resources described in a file allows an institution to practice detailed resource inventory management. Resources can be given names or tags that allow them to be tracked for purposes of cost accounting or performance analysis, allowing an institution to easily group and identify resources, and to identify spending on resources that are being underutilized or not utilized at all. This type of management also aids in process and resource documentation; each resource can be annotated within the code to link to troubleshooting documents, provide human-readable descriptions for processes, and explain the business and technical decisions pertinent to the infrastructure. Additionally, IaC can make it easy for resources to be audited for security considerations, as security and logging mechanisms can be described within the code.

While the resource management aspects of IaC are certainly valuable, it is the automation aspects of this practice that are perhaps its greatest strength. The *code* in Infrastructure as Code (accurately) implies that cloud resources can be created and driven by code rather than human intervention. Enterprise-wide cloud infrastructures can be created entirely within code, and this is also true for discrete applications such as digital repositories. Templates can be created to describe common resources that are to be applied in numerous places throughout a cloud infrastructure. For example, to ensure that a repository conforms to hardware standards, capabilities, and best practices, a template can define the specifications and customizations of the cloud server on which it is to run. This can include processor speed of a virtual server, encryption keys for storage volumes, an operating system version, and any number of other resource customizations.

## Logging and Analytics

A digital repository is a platform for collecting and disseminating information. It makes specific sets of information available to users at remote locations, makes the information discoverable through searching and navigation, presents it through a web browser or stand-alone application, and provides users text, audio, and graphical information through digital means. All of this is done by means of computer hardware and software processes, processes that, due to their digital nature, can be configured to output a detailed record of their use and activity in a collection of files known as logs. Logs are simply files that list the activities occurring on sites, applications, and servers. They can appear as individual files located on servers, or they can be integrated with special logging services as part of a cloud services suite.

The software used to run a repository may generate several types of logs. It is possible that a repository will run on widely adopted web server software, such as Apache or Nginx. These servers (and those like them) generate logs that contain data such as a record of which pages and files are accessed, server health and status, and any server errors that may be occurring behind. It is also possible that a repository is using a database or a search index, both of which may have logging capabilities that could be used to discover what search terms users are entering into the repository search fields. Furthermore, repository software may have custom logging specifically to diagnose errors in application code or to debug the development of add-ons and extensions.

There is some general terminology that should be understood in order to begin interpreting the data being generated by a repository. *Analytics* is "information resulting from the systematic analysis of data or statistics," in this case being from a library or repository.[2] It can be thought of as the practical interpretation or human-readable form of the data that is captured in logs, although analytics themselves can be complex and potentially difficult to understand. Analytics are often displayed in tables or with visual aids that may make them easier to interpret. Analytics are comprised of *metrics*, with a *metric* being one specific type of data. Page views, time spent in the repository, the country that patrons are visiting from—these and more are all metrics that, when analyzed, provide an institution with a meaningful set of information from which business decisions can be made. Along these lines of business intelligence is the term *key performance indicators*, or KPIs. KPIs are measurable values that can demonstrate how successful an institution is at meeting business objectives. They can be tied to goals that an institution has set for itself or its repository; for example, an institution may have KPIs relating to increased page views and increased time spent on each page as a way of illustrating that repository use is increasing.

Cloud technologies can be used to derive analytics from a repository. Most notably, popular cloud-based analytics platforms can offer an institution a simple, comprehensive site or dashboard that can contain popular metrics and visualizations that make analytics convenient to find, readily available, and customizable in their layout and detail. Often these analytics can be used to drive business decisions, such as studying the user base of a repository to better understand who is using the repository and why it is accessed. Additionally, these analytics can offer information as to peak usage hours and the method of access, such as preferred web browser. This information can enable an institution to make changes to a cloud architecture, such as adding processing power during expected periods of high usage or optimizing code to better perform on the web browser most commonly used to access the repository.

## API

Even with the introduction of advanced tools for conducting research, many users are comfortable using basic search techniques to locate resources in a digital repository. This is not necessarily a bad thing; most needs can be met by using simple search boxes and result lists. But it is now possible to use other means to retrieve data from a repository, means that enable large-scale exports or machine-actionable queries, with data returned in a format that can easily be parsed, prepared, and presented by programming languages. One of these means is known as an application programming interface, or API for short. There are different types of APIs, but a common type is known as a restful API. These function by using a URL that can be accessed like any other URL, but instead of directing a user to a website, structured data is returned to the user (these URLs are known as endpoints). This type of advanced retrieval can allow for dynamically generated site content and integration with other sites that share data in the same way. For a repository, this offers a lot of opportunity. For example, a repository may expose all the items in its collection through an API endpoint. When items are exposed like this, a library website can automatically gather this list of items, format them to match its own branding, and publish them on a web page in real time. Alternately, a repository may consume data through an external API. Consider a page in a repository highlighting the achievements of a notable historical figure. The repository could call on an external API that offers publications attributed to this figure, take the data, and publish it in line with related repository content.

In terms of cloud resources, some services provide tools or a framework with which an experienced developer can create and manage an API. In doing so, data for a repository can be exposed though an API that adheres to common standards, meaning that other institutions or developers can use the API in a standard way and achieve the expected results. These cloud-based APIs may tie into the billing, analytics, and security functionality that many service providers integrate across their platforms. In some cases, repositories offered as SaaS applications provide native API functionality as well, making it simple for institutions to offer data through an API without having to create one.

From a systems administration standpoint, some cloud service providers allow users to interact directly with their services using an API. The resources they provide can be managed completely through the API; that is, resources can be created, modified, and deleted by sending commands to an API endpoint. This type of interface access helps to facilitate managing infrastructure as code, as it allows resource management to be automated within scripts or a custom interface.

Another type of advanced data integration is an SDK, or a software development kit. An SDK is not something a basic user might ever use, but when used by a software developer it can enable new features and customizations for repository software. An SDK is a library of computer code, tightly integrated with one or more programming languages, designed to perform specific functions. In the case of a repository, an SDK may provide access to records, search features, site appearance manipulation, or any number of core functions that could aid in developing extensions and integrations to the software. An SDK may provide all the same functionality that an API provides, although some may offer even more granular access to cloud functionality, as they are created with software development in mind. SDKs also commonly include documentation meant specifically for developers to aid in their work.

## Updates

### Infrastructure Updates and Upgrades

Due to their strictly virtual nature, cloud resources can be managed in different ways than physical devices. Notably, they can be created and destroyed in a matter of seconds. While this might sound disastrous, this offers flexibility to an infrastructure that is not feasible for most on-prem IT environments; old, outdated, and insecure resources can be rapidly decommissioned and recommissioned in a fully patched and updated state. As part of automated deployment and maintenance workflows, this method of resource management reduces the need for human interaction, maintains uniformity across resources, and clearly defines the exact setup of every system in the infrastructure. While cloud-based systems administration can be crucial for overall infrastructure maintenance, it can also be useful for a discrete project such as a repository; software updates, patches, and the compatibility of any underlying servers or databases can be handled remotely and reliably with potentially little downtime.

Cloud service providers may offer specific tools for managing infrastructure, and third parties offer these services as SaaS subscriptions. Some of these services exist both inside and outside of cloud computing and are designed to handle common system administration functions like upgrading operating systems or managing the software installed on remote workstations. Other tools are tightly integrated into a larger suite of cloud services, providing an efficient, fast, and interconnected means of managing cloud-specific resources. These tools can provide enhanced automation and advanced capabilities such as scheduled maintenance windows, unified logging, integrated documentation, version tracking, and more.

### Product Enhancements and Extensibility

In addition to handling security and stability updates, cloud services such as SaaS programs can make extending the functionality of a repository a relatively simple process. Adding new features can be as easy as checking a box on an order form, providing some simple configurations, and pressing Download. These extensions may be free additions, or they may come at a cost, and because they are packaged services, they can receive updates and fixes directly from the provider. This may be an optimal workflow for institutions that wish to use only the most common features with their repository or those that prioritize ease of maintenance over design flexibility. SaaS programs are not the only cloud resource to provide benefits to this software extensibility. Cloud-centric software development tools can simplify the process of adding locally developed add-ons and extensions to a repository project. And because cloud systems can be more easily updated than their physical counterparts, new features can be added without worrying that existing servers are not powerful enough to handle the increased capabilities.

## Preservation

Preservation is an important consideration when designing a repository; projects such as these are often the culmination of great amounts of time, effort, and cost. Additionally, these projects may be an important (if not the only) means by which important cultural artifacts are preserved. Being completely virtual in construction, a cloud-based repository may feel decidedly impermanent, as all its content may have never existed anywhere other than the remote network of data centers used by a cloud service provider. Still, there are ways to preserve digital content indefinitely, so that like physical resources, it can be discovered, viewed, and studied for generations to come.

### Permalinks and URL Management

Maintaining discoverability is an important aspect of digital preservation. Search engines, browsers, and online bookmarking services can change or disappear, sometimes with little to no notice, leaving users without an easy path to sites and service. It is difficult to know how every user will navigate to a repository or the individual records therein, but steps can be taken to increase the likelihood that the repository content can remain discoverable over time. One way to do this is to use what are known as permalinks. Permalinks are URLs that are intended to last over time; to provide a continuous, direct link to their destinations; and to avoid the problem of linkrot, or a URL that eventually directs to nothing. Permalinks are often designed to be short and easy to remember and type, instead of being the long, dynamically generated strings of letters and numbers characteristic of many websites and repositories. They can make it simple for users to navigate back to records of interest and can provide some reassurance that items can be easily visited again in the future. Some cloud services offer tools to help manage URLs, including permalinks, URL shortening features, redirects, and dynamic domain mapping.

### Checksums and Other Preservation Tools

For a repository to be considered a trusted and reliable mechanism for storing and accessing digital objects, there must be a way to be sure that the digital objects stay free of errors and corruption. There are tools that can be used to verify that the digital objects stored in a repository have maintained their data integrity and have persisted without developing errors, an idea known as fixity. According the Digital Preservation Coalition, "Fixity could be applied to images or video inside an audiovisual object, to individual files within a zip, to metadata inside an XML structure, to records in a database, or to objects in an object store," all of which may be present in a repository.[3] A specific type of fixity check that can be made is called a checksum. Checksums can be used to validate a digital file and ensure it has not changed over time. Technically speaking, a checksum is a special number or string of characters derived from a formula. Because all digital objects (everything on computers, in fact) actually exists as computer code, this checksum formula can be run on the object to produce one of these special strings of characters. This string is saved (potentially in the repository) for later comparison. At some point later, either as part of a routine check or if file corruption is suspected, the same checksum function is run on the same digital object, which produces another string of characters. If the two strings are the same, this shows that the file has not changed. If they are different, the file has changed in some way, and the change could be a sign of corruption.

### Backups

It goes without saying that backups provide an effective countermeasure against data loss. By copying databases, servers, and other cloud configurations to separate locations, a repository can recover from several failure types and be restored to a functional condition in a short amount of time. Traditional backups have included the simple backup of data, where data is exported from databases and servers on a scheduled or ad hoc basis to be reimported in the recovery process. In a cloud environment, many service providers offer integrated backup features that both capture data automatically and provide a built-in mechanism to recreate resources based on these backups.

If possible, consider backing up content to multiple locations. Some cloud service providers offer backup services to various physical locations while remaining a part of the larger services package. That is to say, within the same cloud software suite, backups can be made in both Washington and Wisconsin, all while still using the same service provider. This can provide a level of convenience while offering some mitigation from failure caused by regional disasters. Using different cloud services for backing up content can further help prevent data loss; in the event of total failure of one of the service providers, backups should still be available on the other service. While perhaps less convenient and more costly, this practice provides an additional level of protection for services where data retention is extremely important.

Thanks to the versatility of cloud resources, it is possible to automate backups of databases and other resources. Backups can be set to a schedule, occurring at a specific date and time. These backups can be given a specific duration as well, causing them to automatically delete after a set amount of time. This can be advantageous because it makes backups available and keeps the storage costs associated with storing backups from growing too large. Another option is tape backups. By saving backups to a tape backup service, physical tapes are created with the backup data. These tapes can outlast any power outage and can be physically moved to a new data center if necessary. Tape backups can be inexpensive and physically durable, although they are not convenient to use and should be considered as a last measure in restoring data.

## Notes

1. Kief Morris, *Infrastructure as Code: Dynamic Systems for the Cloud Age*, 2nd ed. (Sebastopol, CA: O'Reilly Media, Inc., 2020).
2. *OED Online*, s.v. "analytics," accessed October 27, 2020.
3. Digital Preservation Coalition, "Fixity and Checksums," in *Digital Preservation Handbook*, 2nd ed. (Glasgow, Scotland: Digital Preservation Coalition, 2015), https://www.dpconline.org/handbook/technical-solutions-and-tools/fixity-and-checksums.