# Migrating On-Premises Hadoop Infrastructure to Google Cloud

This guide provides an overview of how to move your on-premises Apache Hadoop system to Google Cloud. It describes a migration process that not only moves your Hadoop work to Google Cloud, but also enables you to adapt your work to take advantage of the benefits of a Hadoop system optimized for cloud computing. It also introduces some fundamental concepts you need to understand to translate your Hadoop configuration to Google Cloud.

This is the first of several guides describing how to move from on-premises Hadoop:

- This guide, which provides context and planning advice for your migration.
- Migrating HDFS Data from On-Premises to Google Cloud (https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-data) provides additional context for incrementally moving your data to Google Cloud.
- Migrating data from HBase to Cloud Bigtable (https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-data-hbase-to-bigtable)
- Migrating Hadoop Jobs from On-Premises to Dataproc (https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-jobs) describes the process of running your jobs on Dataproc and other Google Cloud products.

## The benefits of migrating to Google Cloud

There are many ways in which using Google Cloud can save you time, money, and effort compared to using an on-premises Hadoop solution. In many cases, adopting a cloud-based approach can make your overall solution simpler and easy to manage.

### Built-in support for Hadoop

Google Cloud includes Dataproc, which is a managed Hadoop and Spark environment. You can use Dataproc to run most of your existing jobs with minimal alteration, so you don't need to move away from all of the Hadoop tools you already know.

### Managed hardware and configuration

When you run Hadoop on Google Cloud, you never need to worry about physical hardware. You specify the configuration of your cluster, and Dataproc allocates resources for you. You can scale your cluster at any time.

### Simplified version management

Keeping open source tools up to date and working together is one of the most complex parts of managing a Hadoop cluster. When you use Dataproc, much of that work is managed for you by Dataproc versioning (https://cloud.google.com/dataproc/docs/concepts/versioning/overview).

### Flexible job configuration

A typical on-premises Hadoop setup uses a single cluster that serves many purposes. When you move to Google Cloud, you can focus on individual tasks, creating as many clusters as you need. This removes much of the complexity of maintaining a single cluster with growing dependencies and software configuration interactions.

## Planning your migration

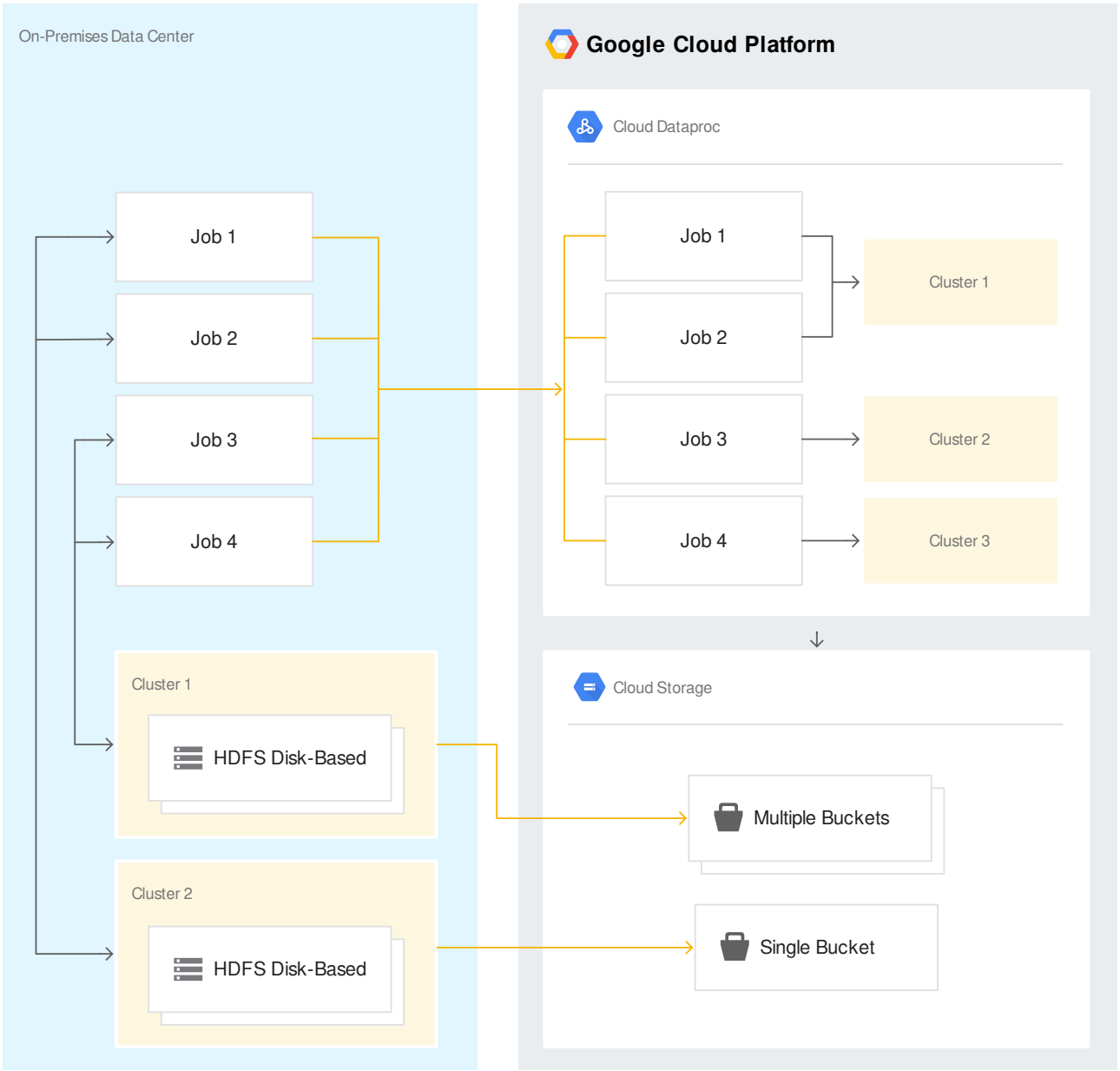
Migrating from an on-premises Hadoop solution to Google Cloud requires a shift in approach. A typical on-premises Hadoop system consists of a monolithic cluster that supports many workloads, often across multiple business areas. As a result, the system becomes more complex over time and can require administrators to make compromises to get everything working in the monolithic cluster. When you move your Hadoop system to Google Cloud, you can reduce the administrative complexity. However, to get that simplification and to get the most efficient processing in Google Cloud with the minimal cost, you need to rethink how to structure your data and jobs.

Because Dataproc runs Hadoop on Google Cloud, using a persistent Dataproc cluster to replicate your on-premises setup might seem like the easiest solution. However, there are some limitations to that approach:

- Keeping your data in a persistent HDFS cluster using Dataproc is more expensive than storing your data in Cloud Storage, which is what we recommend, as explained later. Keeping data in an HDFS cluster also limits your ability to use your data with other Google Cloud products.
- Augmenting or replacing some of your open-source-based tools with other related Google Cloud services can be more efficient or economical for particular use cases.
- Using a single, persistent Dataproc cluster for your jobs is more difficult to manage than shifting to targeted clusters that serve individual jobs or job areas.

The most cost-effective and flexible way to migrate your Hadoop system to Google Cloud is to shift away from thinking in terms of large, multi-purpose, persistent clusters and instead think about small, short-lived clusters that are designed to run specific jobs. You store your data in Cloud Storage to support multiple, temporary processing clusters. This model is often called the *ephemeral model*, because the clusters you use for processing jobs are allocated as needed and are released as jobs finish.

The following diagram shows a hypothetical migration from an on-premises system to an ephemeral model on Google Cloud.




The example moves four jobs that run on two on-premises clusters to Dataproc. The ephemeral clusters that are used to run the jobs in Google Cloud are defined to maximize efficiency for individual jobs. The first two jobs use the same cluster, while the third and fourth jobs each run on their own cluster. When you migrate your own jobs, you can customize and optimize clusters for individual jobs or for groups of jobs as makes sense for your specific work. Dataproc helps you quickly define multiple clusters, bring them online, and scale them to suit your needs.

The data in the example is moved from two on-premises HDFS clusters to Cloud Storage buckets. The data in the first cluster is divided among multiple buckets, and the second cluster

is moved to a single bucket. You can customize the structure of your data in Cloud Storage to suit the needs of your applications and your business.

The example migration captures the beginning and ending states of a complete migration to Google Cloud. This implies a single step, but you'll get the best results if you don't think of moving to Google Cloud as a one-time, complete migration. Instead, think of it as refactoring your solutions to use a new set of tools in ways that weren't possible on-premises. To make such a refactoring work, we recommend migrating incrementally.

## Recommended sequence for migrating to Google Cloud

Here are the recommended steps for migrating your workflows to Google Cloud:

1. Move your data first
    - Move your data into Cloud Storage buckets.
    - Start small. Use backup or archived data to minimize the impact to your existing Hadoop system.
2. Experiment
    - Use a subset of data to test and experiment. Make a small-scale proof of concept for each of your jobs.
    - Try new approaches to working with your data.
    - Adjust to Google Cloud and cloud-computing paradigms.
3. Think in terms of specialized, ephemeral clusters.
    - Use the smallest clusters you can—scope them to single jobs or small groups of closely related jobs.
    - Create clusters each time you need them for a job and delete them when you're done.
4. Use Google Cloud tools wherever appropriate.

## Moving to an ephemeral model

The biggest shift in your approach between running an on-premises Hadoop workflow and running the same workflow on Google Cloud is the shift away from monolithic, persistent clusters to specialized, ephemeral clusters. You spin up a cluster when you need to run a job

and then delete it when the job completes. The resources required by your jobs are active only when they're being used, so you only pay for what you use. This approach enables you to tailor cluster configurations for individual jobs. Because you aren't maintaining and configuring a persistent cluster, you reduce the costs of resource use and cluster administration.

This section describes how to move your existing Hadoop infrastructure to an ephemeral model.

### Separate data from computation

Using Cloud Storage as the persistent storage for your workflows has the following advantages:

- It's a Hadoop Compatible File System (HCFS), so it's easy to use with your existing jobs.
- It's faster than HDFS in many cases.
- It requires less maintenance than HDFS.
- It enables you to easily use your data with the whole range of Google Cloud products.
- It's considerably less expensive than keeping your data in HDFS on a persistent Dataproc cluster.

With your data stored persistently in Cloud Storage, you can run your jobs on ephemeral Hadoop clusters managed by Dataproc.

In some cases, it might make more sense to move data to another Google Cloud technology, such as BigQuery or Cloud Bigtable. But most general-purpose data should persist in Cloud Storage. More detail about these alternative storage options is provided later in this guide.

### Run jobs on ephemeral clusters

Dataproc makes it easy to create and delete clusters so that you can move away from using one monolithic cluster to using many ephemeral clusters. This approach has several advantages:

- You can use different cluster configurations for individual jobs, eliminating the administrative burden of managing tools across jobs.
- You can scale clusters to suit individual jobs or groups of jobs.
- You only pay for resources when your jobs are using them.

- You don't need to maintain clusters over time, because they are freshly configured every time you use them.
- You don't need to maintain separate infrastructure for development, testing, and production. You can use the same definitions to create as many different versions of a cluster as you need when you need them.

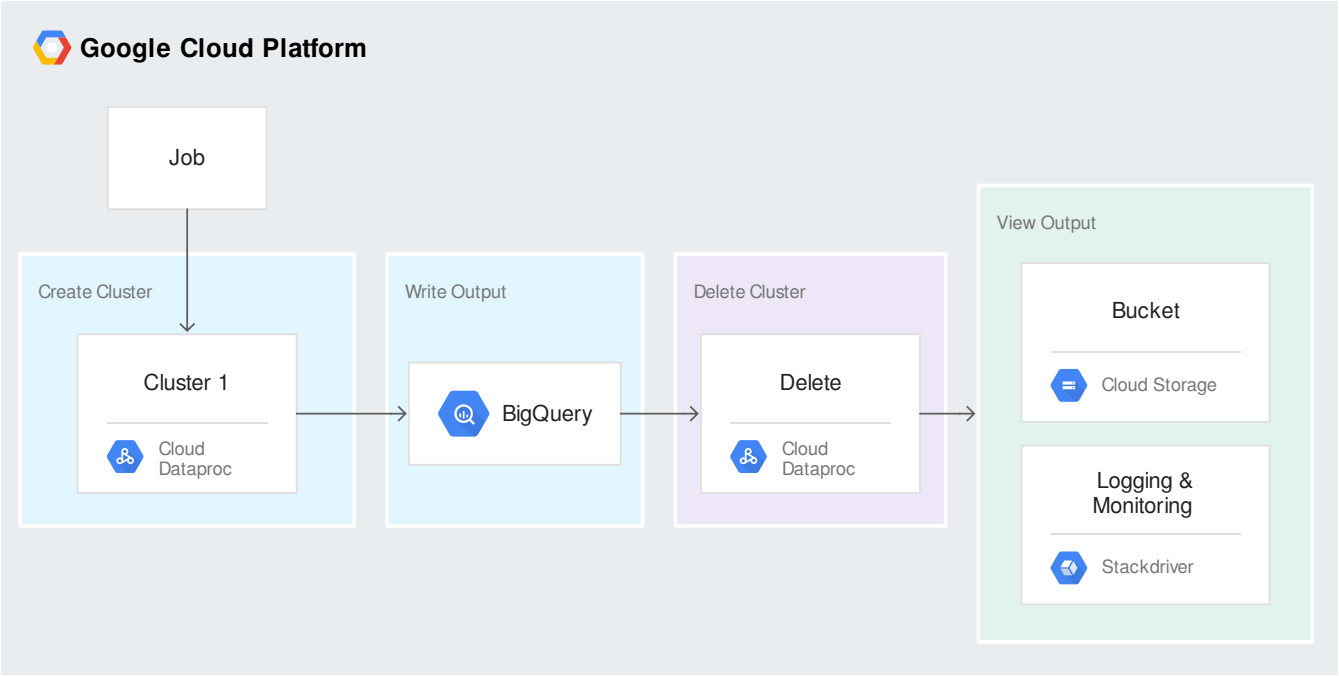
You can use Dataproc initialization actions (https://github.com/GoogleCloudDataproc/initialization-actions) to define the configuration of nodes in a cluster. This makes it easy to maintain the different cluster configurations you need to closely support individual jobs and related groups of jobs. You can use the provided sample initialization actions (https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/init-actions) to get started. The samples demonstrate how to make your own initialization actions.

### Minimize the lifetime of ephemeral clusters

The point of ephemeral clusters is to use them only for the jobs' lifetime. When it's time to run a job, follow this process:

1. Create a properly configured cluster.
2. Run your job, sending output to Cloud Storage or another persistent location.
3. Delete the cluster.
4. Use your job output however you need to.
5. View logs in Stackdriver or Cloud Storage.

This process is shown in the following diagram:




**Use small persistent clusters only when absolutely necessary**

If you can't accomplish your work without a persistent cluster, you can create one. This option may be costly and isn't recommended if there is a way to get your job done on ephemeral clusters.

You can minimize the cost of a persistent cluster by:

- Creating the smallest cluster you can.
- Scoping your work on that cluster to the smallest possible number of jobs.
- Scaling the cluster to the minimum workable number of nodes, adding more dynamically to meet demand.

## Migrating incrementally

There are many advantages to migrating incrementally. You can:

- Isolate individual jobs in your existing Hadoop infrastructure from the complexity that's inherent in a mature environment.
- Examine each job in isolation to evaluate its needs and to determine the best path for migration.

- Handle unexpected problems as they arise without delaying dependent tasks.
- Create a proof of concept for each complex process without affecting your production environment.
- Move your workloads to the recommended ephemeral model thoughtfully and deliberately.

Your migration is unique to your Hadoop environment, so there is no universal plan that fits all migration scenarios. Make a plan for your migration that gives you the freedom to translate each piece to a cloud-computing paradigm.

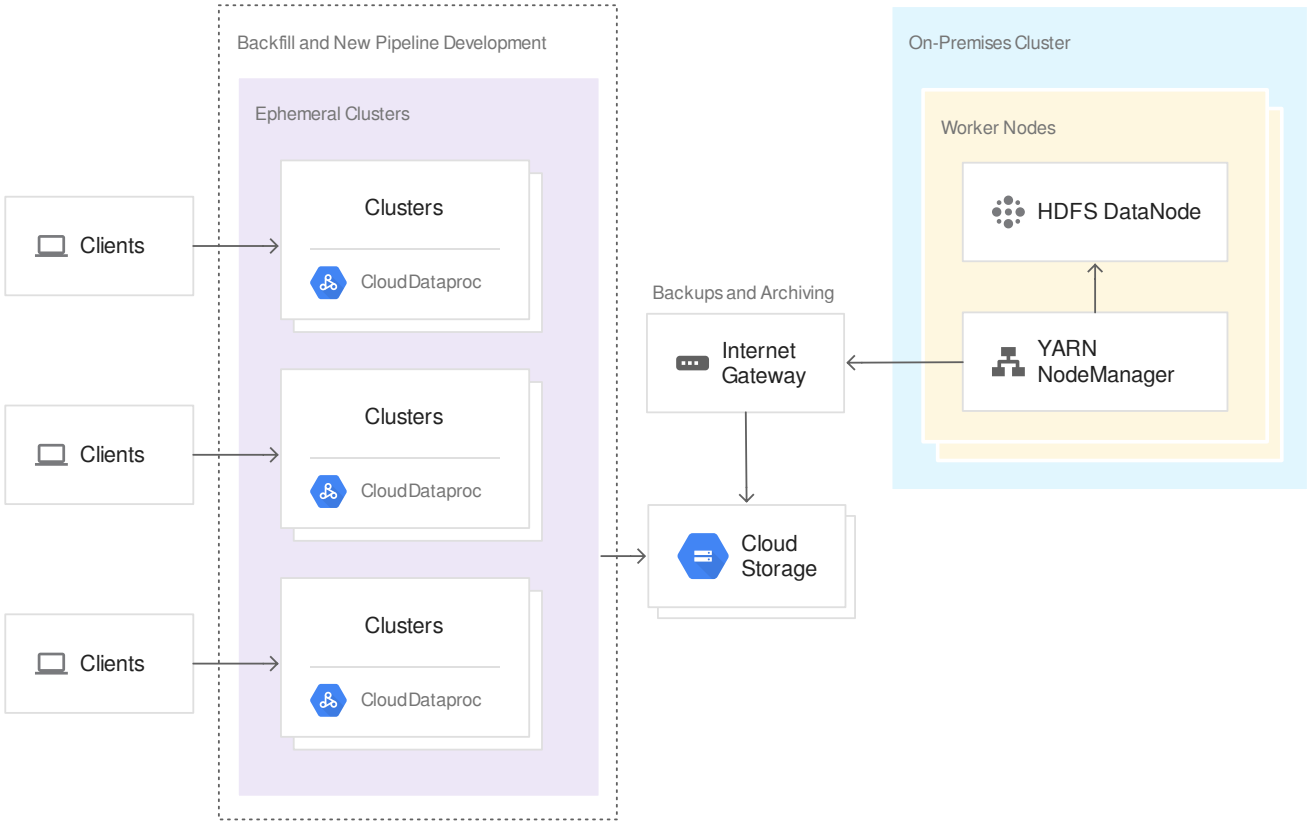
Here's a typical incremental migration sequence:

1. Move some portion of your data to Google Cloud.
2. Experiment with that data:
    a. Replicate your existing jobs that use the data.
    b. Make new prototypes that work with the data.
3. Repeat with additional data.

Start with the least critical data that you can. In the early stages, it's a good approach to use backup data and archives.

One kind of lower-risk job that makes a good starting test is backfilling by running burst processing on archive data. You can set up jobs that fill gaps in processing for data that existed before your current jobs were in place. Starting with burst jobs can often provide experience with scaling on Google Cloud early in your migration plan. This can help you when you begin migrating more critical jobs.

The following diagram shows an example of a typical backfill hybrid architecture.




The example has two major components. First, scheduled jobs running on the on-premises cluster push data to Cloud Storage through an internet gateway. Second, backfill jobs run on ephemeral Dataproc clusters. In addition to backfilling, you can use ephemeral clusters in Google Cloud for experimentation and creating proofs of concept for future work.

## Planning with the completed migration in mind

So far, this guide has assumed that your goal is to move your entire Hadoop system from on-premises to Google Cloud. A Hadoop system running entirely on Google Cloud is easier to manage than one that operates both in the cloud and on-premises. However, a hybrid approach is often necessary to meet your business or technological needs.

### Designing a hybrid solution

Here are some reasons you might need a hybrid solution:

- You are in the process of developing cloud-native systems, so existing systems that depend on them must continue to run on-premises until you are finished.

- You have business requirements to keep your data on-premises.
- You have to share data with other systems that run on-premises, and those systems can't interact with Google Cloud due to technical or business restrictions.

A typical hybrid solution has four major parts:

1. An on-premises Hadoop cluster.
2. A connection from the on-premises cluster to Google Cloud.
3. Centralized data storage in Google Cloud.
4. Cloud-native components that work on data in Google Cloud.

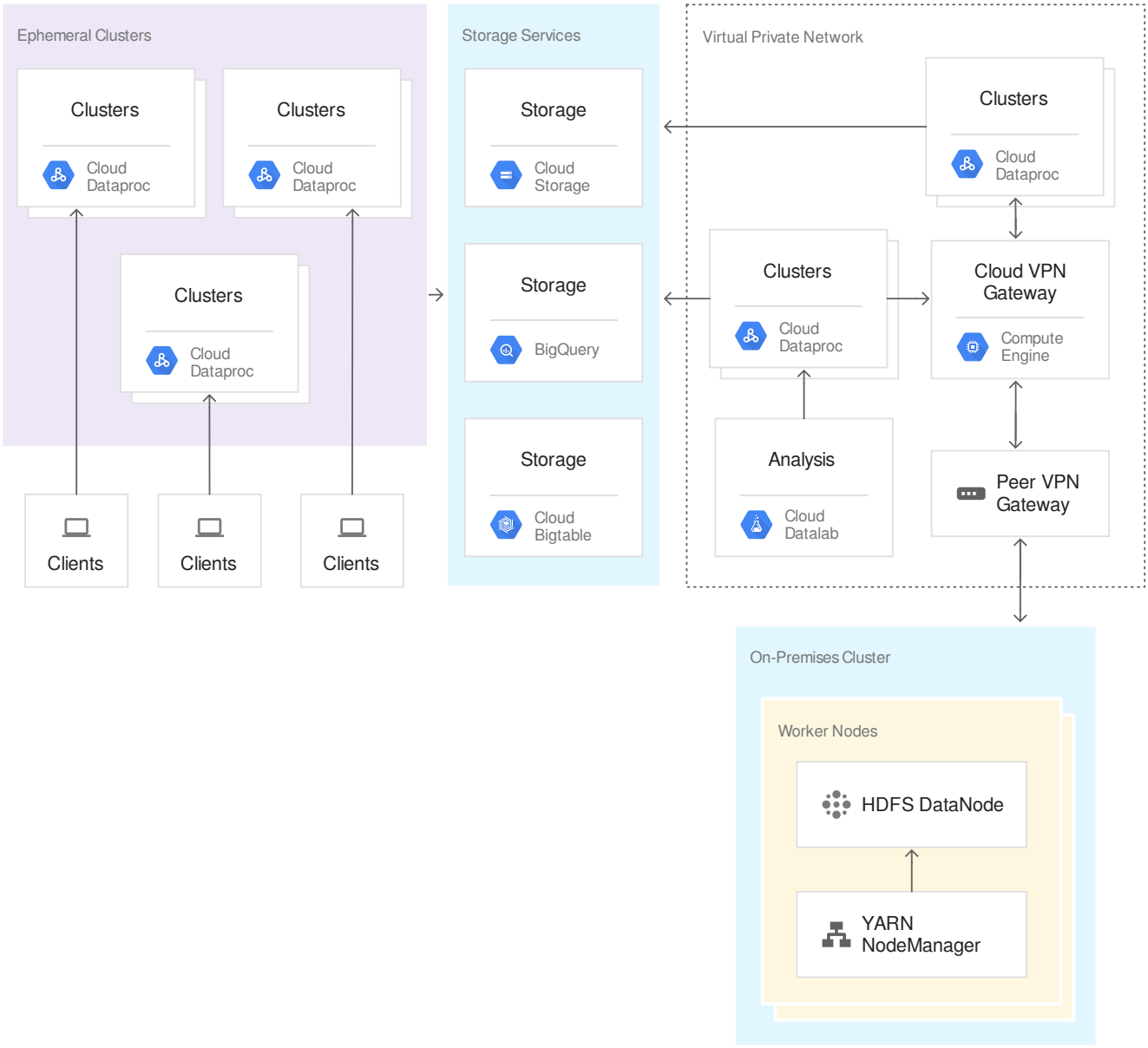
An issue you must address with a hybrid cloud solution is how to keep your systems in sync. That is, how will you make sure that changes you make to data in one place are reflected in the other? You can simplify synchronization by making clear distinctions between how your data is used in the different environments.

For example, you might have a hybrid solution where only your archived data is stored in Google Cloud. You can set up scheduled jobs to move your data from the on-premises cluster to Google Cloud when the data reaches a specified age. Then you can set up all of your jobs that work on the archived data in Google Cloud so that you never need to synchronize changes to your on-premises clusters.

Another way to divide your system is to move all of the data and work for a specific project or working group to Google Cloud while keeping other work on premises. Then you can focus on your jobs instead of creating complex data synchronization systems.

You might have security or logistical concerns that complicate how you connect your on-premises cluster to Google Cloud. One solution is to use a Virtual Private Cloud (https://cloud.google.com/vpc/docs/) connected to your on-premises network using Cloud VPN (https://cloud.google.com/vpn/docs/).

The following diagram shows an example hybrid cloud setup:




The example setup uses Cloud VPN to connect a Google Cloud VPC to an on-premises cluster. The system uses Dataproc inside the VPC to manage persistent clusters that process data coming from the on-premises system. This might involve synchronizing data between the systems. Those persistent Dataproc clusters also transfer data coming from the on-premises system to the appropriate storage services in Google Cloud. For the sake of illustration, the example uses Cloud Storage, BigQuery, and Bigtable for storage—those are the most common destinations for data processed by Hadoop workloads in Google Cloud.

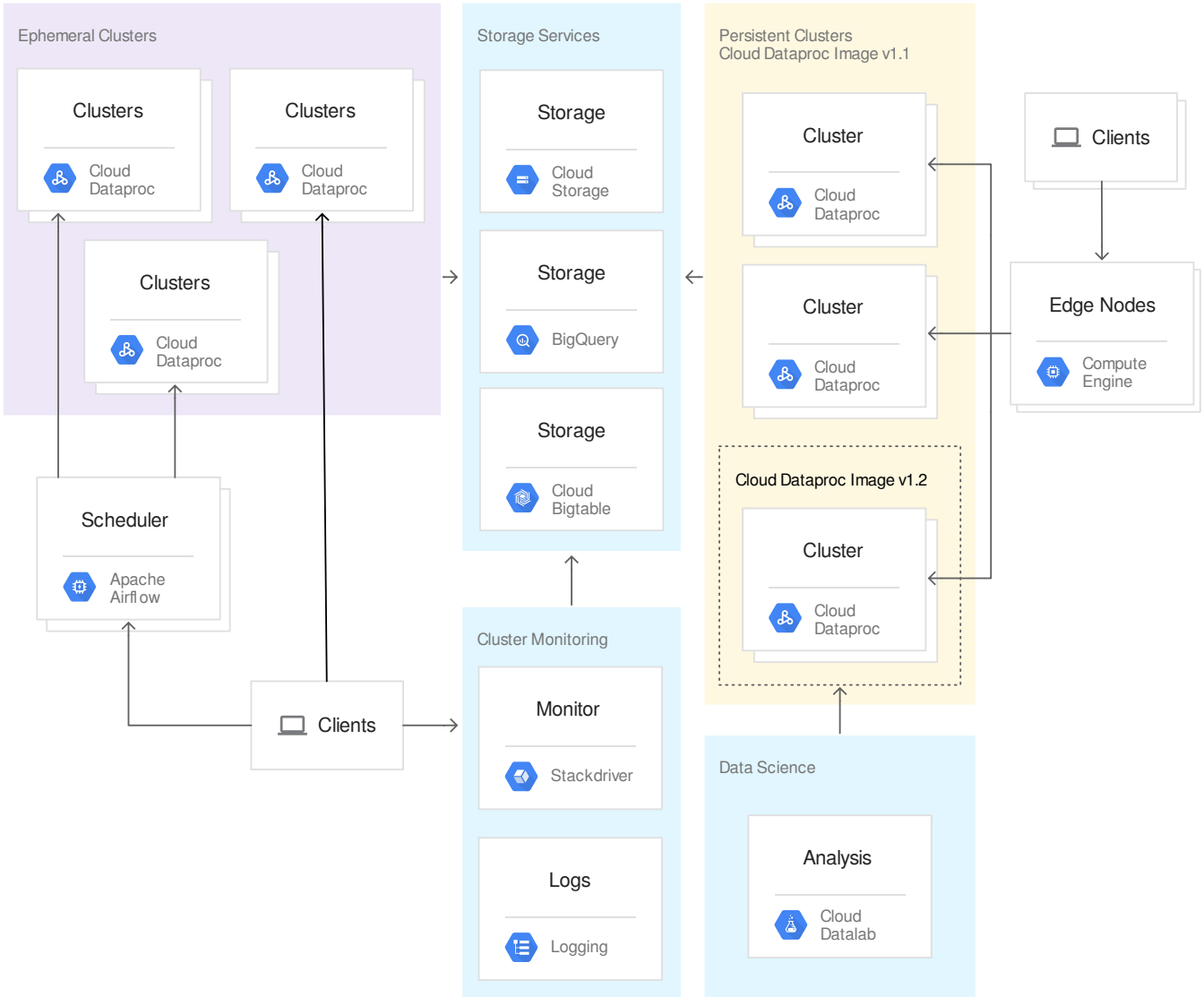
The other half of the example solution shows multiple ephemeral clusters that are created as needed in the public cloud. Those clusters can be used for many tasks, including those that

collect and transform new data. The results of these jobs are stored in the same storage services used by the clusters running in the VPC.

### Designing a cloud-native solution

By contrast, a cloud-native solution is straightforward. Because you run all of your jobs on Google Cloud using data stored in Cloud Storage, you avoid the complexity of data synchronization altogether, although you must still be careful about how your different jobs use the same data.

The following diagram shows an example of a cloud-native system:



The example system has some persistent clusters and some ephemeral ones. Both kinds of clusters share cloud tools and resources, including storage and monitoring. Dataproc uses standardized machine images (https://cloud.google.com/dataproc/docs/concepts/versioning/dataproc-versions) to define software configurations on VMs in a cluster. You can use these predefined images as the basis for the VM configuration you need. The example shows most of the persistent clusters running on version 1.1, with one cluster running on version 1.2. You can create new clusters with customized VM instances whenever you need them. This lets you isolate testing and development environments from critical production data and jobs.

The ephemeral clusters in the example run a variety of jobs. This example shows Apache Airflow (https://airflow.apache.org/) running on Compute Engine (https://cloud.google.com/compute/docs) being used to schedule work with ephemeral clusters.

## Working with Google Cloud services

This section discusses some additional considerations for migrating Hadoop to Google Cloud.

### Replacing open source tools with Google Cloud services

Google Cloud offers many products that you can use with your Hadoop system. Using a Google Cloud product can often have benefits over running the equivalent open source product in Google Cloud. Learn about Google Cloud products and services (https://cloud.google.com/docs) to discover the breadth of what the platform has to offer.

### Using regions and zones

You should understand the repercussions of geography and regions (https://cloud.google.com/docs/geography-and-regions) before you configure your data and jobs. Many Google Cloud services require you to specify regions or zones in which to allocate resources. The latency of requests can increase when the requests are made from a different region than the one where the resources are stored. Additionally, if the service's resources and your persistent data are located in different regions, some calls to Google Cloud services might copy all of the required data from one zone to another before processing. This can have a severe impact on performance.

## Configuring authentication and permissions

Your control over permissions in Google Cloud services is likely to be less fine-grained than you are accustomed to in your on-premises Hadoop environment. Make sure you understand how access management works in Google Cloud before you begin your migration.

Cloud Identity and Access Management (Cloud IAM) (https://cloud.google.com/iam/docs/) manages access to cloud resources. It works on the basis of accounts and roles. Accounts identify a user or request (authentication), and the roles granted to an account dictate the level of access (authorization). Most Google Cloud services provide their own set of roles to help you fine-tune permissions. As part of the planning process for your migration, you should learn about how Cloud IAM interacts with Cloud Storage (https://cloud.google.com/storage/docs/access-control/using-iam-permissions) and with Dataproc (https://cloud.google.com/dataproc/docs/concepts/iam/iam). Learn about the permissions models of each additional Google Cloud service as you add it to your system, and consider how to define roles that work across the services that you use.

## Monitoring jobs with Stackdriver Logging

Your Google Cloud jobs send their logs to Logging (https://cloud.google.com/logging/docs), where the logs are easily accessible. You can get your logs in these ways:

- With a browser-based graphical interface using Logs Viewer (https://console.cloud.google.com/logs/) in Google Cloud Console.
- From a local terminal window using the gcloud command-line tool (https://cloud.google.com/sdk/gcloud/reference/logging/).
- From scripts or applications using the Cloud Client Libraries for the Stackdriver Logging API (https://cloud.google.com/logging/docs/reference/libraries).
- By calling the Logging REST API (https://cloud.google.com/logging/docs/reference/v2/rest/).

## Managing your edge nodes with Compute Engine

You can use Compute Engine (https://cloud.google.com/compute/docs) to access an edge node in a Dataproc Hadoop cluster. As with most Google Cloud products, you have multiple options for management: through the web-based console, from the command line, and through web APIs.

### Using Google Cloud big data services

Cloud Storage is the primary way to store unstructured data in Google Cloud, but it isn't the only storage option. Some of your data might be better suited to storage in products designed explicitly for big data.

You can use Bigtable (https://cloud.google.com/bigtable/docs/overview) to store large amounts of sparse data. Bigtable is an HBase-compliant API that offers low latency and high scalability to adapt to your jobs.

For data warehousing, you can use BigQuery (https://cloud.google.com/bigquery/docs).

## What's next

- Check out the other parts of the Hadoop migration guide:
  - Data migration guide (https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-data)
  - Job migration guide (https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-jobs)
- Explore the Dataproc documentation (https://cloud.google.com/dataproc/docs/).
- Get an overview of Google Cloud (https://cloud.google.com/docs/overview).
- Try out other Google Cloud features for yourself. Have a look at our tutorials (https://cloud.google.com/docs/tutorials).

---

*Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License* (https://creativecommons.org/licenses/by/4.0/)*, and code samples are licensed under the Apache 2.0 License* (https://www.apache.org/licenses/LICENSE-2.0)*. For details, see our Site Policies* (https://developers.google.com/terms/site-policies)*. Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated January 16, 2020.*