The Well-Architected guide to S3

Welcome to the first book in the “Well-Architected Guide” series, which brings you a collection of features and articles created for this series and some adapted from the AWS documentation and blogs. I’d like to say a big thank you to all who have supported me whilst I’ve created this completion of guides and howto’s that help you use S3 in a Well-Architected way. We’ll work our way through the 6 pillars of the well-architected framework with recipes you can try out for yourself to help your standing when scoring your AWS Well-Architected Review.

Overview

Let’s get started!

Who is this book for?

This book is for people who know the basics of AWS and Amazon S3 but want to make sure they are using it in the best possible way, a way that adheres to the six pillars of the well-architected framework of best practices. The book is split into the pillars of the well architected framework so you can jump to Security if you want to check out those reciepes. There are a few tips and tricks in here even for the more experienced practitioners. I’ll guide you through the best practices for security and show you how to save money on common tasks. We’ll look at static web hosting, offloading static assets from your CMS and much much more as we go. One of my favourite integrations with S3 is the ability to automate actions after a file has been uploaded with Lambda.

What can you get from this book?

This book will guide you round some of the best practices around using Amazon S3 and hopefully help you discover it’s much much more than just a simple storage service, I’m a self-confessed S3 fanboy and hopefully this book will share my enthusiasm with you. You’ll learn how to secure your workloads and host a website that can take an impressive amount of load!

Why Well-Architected?

The Well-Architected Framework provides best practices in a business and technical way with regards to your cloud deployment. Keeping the balance of delivering the best and most relevant technology with observability will help you run a smooth experience for your customers.

It’s a set of design principles that can be broadly applied to most clouds and on-premises deployments. However, we tend to think of it as an AWS thing but it’s so much more. We first saw the framework as a set of white-papers in 2015 and like you are now I was skeptical, I think we’ve all read some pretty dull white-papers in our time. The Well-Architected Framework, however, was a refreshing read and I must admit I was an instant fanboy. I was lucky enough to work as a customer and alongside one of its creators throughout my career and have truly valued its impact on the industry and allowed me to do the right thing for my customers in a validated way. Applying these pillars to projects ensures a successful release of a workload every time. Since its launch there have been modifications and updates, there is now a Well-Architected tool in the console where you answer questions to qualify your workload against the pillars and identify areas that may need improvement, but once you get going you'll find it handy to validate your work and have the tools and docs to prove it. There are also lenses for Machine Learning, Data Analytics, Serverless and many more. If you want to check it out in more detail head here: https://aws.amazon.com/architecture/. The next few chapters collect together recipes that broadly fit into one of the six pillars of the Well-Architected Framework, these are:

- Opperational Excellence
- Security
- Reliability
- Performanace Efficiency
- Cost Optimization
- Sustainability

I've marked each recipe with a primary pillar (explained in the concepts section). However the pillars often cross over so i've highlighted when doing a certain action may affect another pillar also. Check out the concepts used in the is book to get a better idea of that.

Concepts

What you need to know!

Whilst reading the sections you'll be presented with the Well Architected Bar, this is to tell you which parts of the Well-Architected Framework is applicape to the content you are about to read. The example below indicates that **Security** is the main focus of the content but the content is also applicable to **Operational Excellence** as a secondary catergory.

OpEx Sec Rel Perf Cost Sus

This book includes code samples and command line (CLI) instructions to run as you go through it. The code blocks will be Terraform or Python code, the code blocks will always be tagged with the correct language like so:

```python
import boto3, botocore

import base64, os, json, requests

from aws_lambda_powertools import Tracer

from aws_lambda_powertools.logging.logger import set_package_logger



set_package_logger()



# POWERTOOLS_SERVICE_NAME defined

tracer = Tracer(service="s3r")



def check_safeurl(url):

    ''' check url to make sure its not on a blocked list'''

    if os.environ['SafeBrowsing'] == 'true':

......
```

Command Line instructions will be indented on their own line, for example

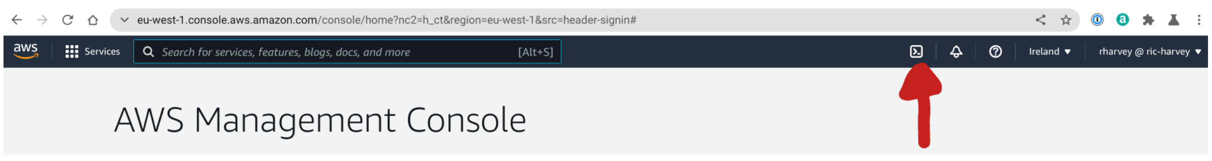
```
aws s3 ls
```

Should you need to change a variable in the code or CLI instruction it will be highlighted in **bold** and **<>** for example **<AWS_REGION>**

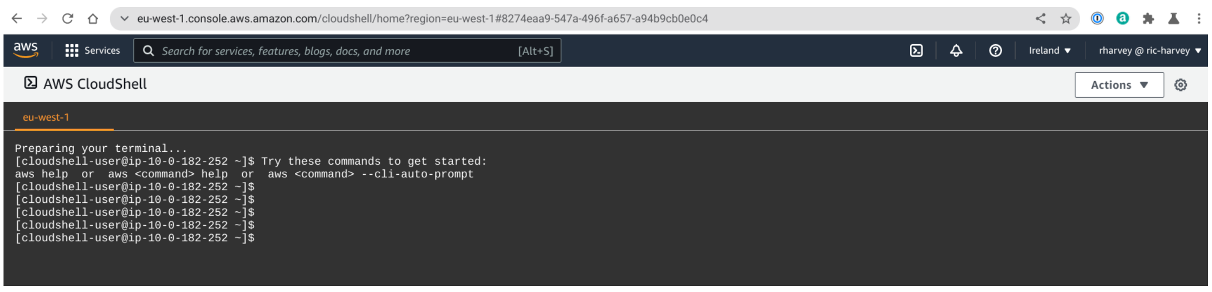
Getting Setup

Getting your environment ready for working with this book
I recommend that unless you have everything ready to go on your local machine that you use the AWS cloud console shell to run the examples in this book. It comes pre-configured with you IAM keys for access to your account and already has the aws CLI and python installed. Be warned python defaults to version 2.7.18 at the time of writing however version 3.7 is installed and that's what we will use.

To open the cloud shell simply click on the shell icon in the top right hand corner once you log into your AWS account:


eu-west-1.console.aws.amazon.com/console/home?nc2=h_ct&region=eu-west-1&src=header-signin#
aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
rharvey @ ric-harvey
AWS Management Console

You should tehn be prompted with the following screen after 30 seconds whilst it spins up:


eu-west-1.console.aws.amazon.com/cloudshell/home?region=eu-west-1#8274eaa9-547a-496f-a657-a94b9cb0e0c4
aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
rharvey @ ric-harvey
AWS CloudShell
Actions
eu-west-1

```
Preparing your terminal...
[cloudshell-user@ip-10-0-182-252 ~]$ Try these commands to get started:
aws help  or  aws <command> help  or  aws <command> --cli-auto-prompt
[cloudshell-user@ip-10-0-182-252 ~]$
[cloudshell-user@ip-10-0-182-252 ~]$
[cloudshell-user@ip-10-0-182-252 ~]$
[cloudshell-user@ip-10-0-182-252 ~]$
[cloudshell-user@ip-10-0-182-252 ~]$
```

I now recommend you run some updates to get us all on the same track:

```
yum update
yum upgrade
```

Installing Terraform

Finally in the setup section lets install terraform. Terraform is a Infrastructure as code tool produced by hashicorp and is an alternative to Amazon's own CloudFormation. I'm including both in this book as which one you prefer is down to your preference.

Let us install unzip first so you can extract the terraform file:

```
sudo yum install unzip
```

Find the latest release version of terraform from here: https://github.com/hashicorp/terraform/releases in my case its 1.1.2

```
export release=<VERSION>
```

This sets the variable $release for the following commands

```
sudo wget
https://releases.hashicorp.com/terraform/${release}/terraform_${release}_linux
_amd64.zip
```

You should see the output of wget downloading the file here. Now lets unzip the file and put it somewhere useful.

```
unzip terraform_${release}_linux_amd64.zip
 sudo unzip terraform_${release}_linux_amd64.zip
 sudo mv terraform /usr/local/bin/
 terraform version
```

The final command should return the current version of terraform, in my case, it’s 1.1.2

What is S3?

What can you do with S3?

Amazon S3 is a Simple Storage Service, it's an object-store. An object store is different from traditional file systems as in it is a flat structure. This is the strangest bit to wrap your head around because when you see files on an object store you often see them in a way that represents folders, don't be fooled it's all flat (unlike the earth). We call the files we place in the store objects and they are comprised of the data and the metadata, the store itself we call a bucket. You access S3 in a multitude of ways, there's an API with SDK's for many languages (python,go,nodeJS,Java,.net, plus many more), there's an SFTP service you can run, console access or you can use 3rd party applications like cyberduck, you can even mount it as a file system with a set of tools we'll talk about later. When using S3 it used to be the case that there was eventual consistency, this came from the fact that when you made an API request to upload a file it took time for all the other endpoints to acknowledge the file in the store, so if you tried a read after write API call it could sometimes fail. You've probably read this on the internet a few times. However, the good news is that it is no longer true, as S3 supports strong consistency and you can do read's immediately after a write these days.

We are going to learn in this book how we can use S3 to:

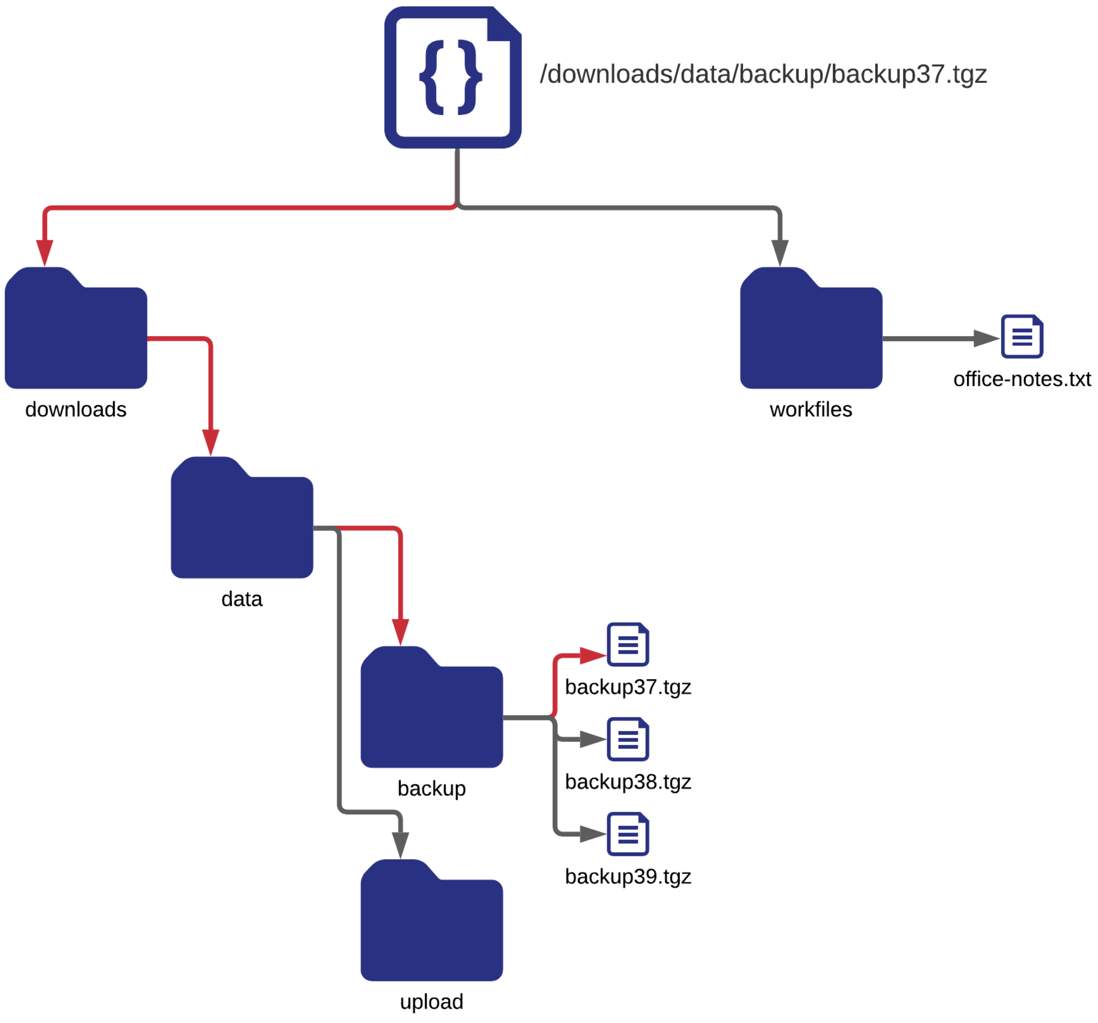
- Secure S3
- Automate creation and lifecycle policies
- Store files (large and small)
- Host simple web sites
- Trigger actions in lambda,SNS and EventBridge

To understand how an object store differs to file storage lets look at how each work.

File Storage

What is file Storage?

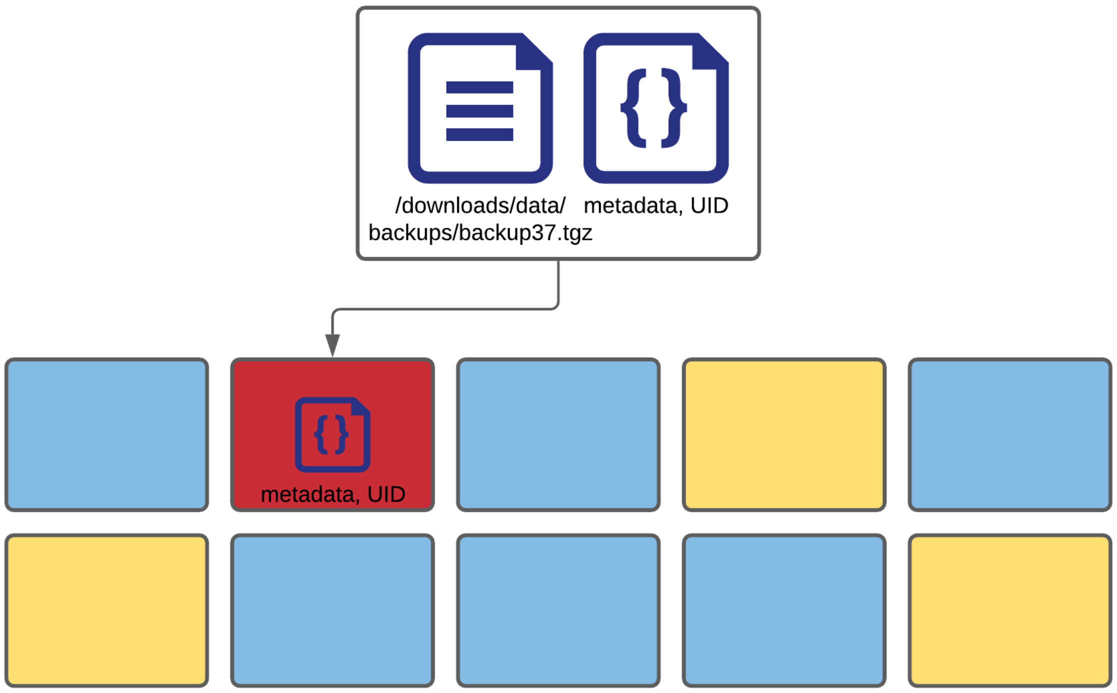
File storage puts data in folder-like structures and a hierarchical file. The file is stored as one file, one chunk of data and is not broken down into smaller blocks. Folders can be nested and in order to retrieve the file, the entire directory path is required to get to the data. Lots of NAS systems you buy at-home use this type of storage because it's cheaper than block storage. It's very good for lots of small files and can potentially scale to millions of files. The file system is generally local, as in one physical location. File storage also stores limited metadata with the file, typically, date created, file, and path. If you think of file storage like a multi-story car park, you drive in and park in a spot. In order to get your car back, you need to remember the car park location, the floor you were parked on, and the space in which you left the car. All of those things are needed to successfully get your car back.



Object Storage

What is Object Storage?

Object storage on the other hand is a flat storage system of unstructured data. The files are stored in whole or part along with metadata and unique identifiers that are used to locate and identify the data you are accessing. The metadata tags have much more detail, such as location, owner, creation timestamp, file type, and many more. The data and the metadata are bundled together and placed into a flat storage pool for retrieval later. It is a low-cost approach to storage allowing systems to massively scale and even be dispersed globally. It typically has a slightly longer latency for retrieval than a file storage system and you generally access it via REST API calls rather than in a POSIX nature (but we'll chat about this later. In our car park analogy this time we drive to the car park and hand the keys to a valet, who in turn gives us a ticket. The car is parked for us in a massive ground-level car park that stretched for miles and when we return the ticket the car is fetched from the right space. S3 is an object so fits this description and it's big, I'll dive into more details in chapter 2.



The main differences between the two are summarized in this table below:

WA Pillar	File Storage	Object Storage
Operational Excellence	A limited number of metadata tags	Customizable metadata tags with no limits
Reliability	Typically on one physical site	Can be scaled globally across multiple regions
Reliability	Scales to millions of files	Scales infinitely beyond petabytes
Performance Efficiency	Best performance with small files	Great for large files and high concurrency and throughput
Cost Optimization	SAN solutions can be expensive	Cheaper and cloud providers only charge for what you use

Object storage is very cost-efficient for many use cases. It also has several tiers of storage available to the end-user. These tiers have pros and cons and you can find out more in chapter 5.

Operational Excellence

Support development and run workloads effectively by gaining insight into operations continuously

This focuses on running and monitoring your deployment and continually evaluating the systems for improvement. It covers a great deal of how you set things up and should cover Infrastructure as code (IaC) as a core practice, as well as how you respond to requests and incidents.

From a business point of view a good solutions architect should be looking at how you fold back the learnings into processes, ensuring leasons are learn when there are operational issues and helping the business either avoid or recover more quickly. You should also be looking at the new feature’s be released and deciding if this will help you remove operational overhead. A great example of this is when intelligent tiering was released. Intelligent tiering in a nut shell helps you manage you data in S3 storage classes and moves the least active objects to cheaper tiers of storage. Before intelligent tiering, devops team would have to monitor this and make decisions on the creation of lifecycle policies, now intelligent tiering is here this free’s up time for the devops team and removes the human error factor also.

The AWS Command Line

Useful AWS CLI commands

OpEx Sec Rel Perf Cost Sus

Let’s start off by getting familiar with the AWS CLI. If you are using the AWS Web Terminal as suggested in getting started this will be super simple. Manipulating large numbers of files is much simpler via the CLI compared to using the console interface, the “‘sync’” command is one example of this allowing you to easily copy a large number of files to or from an S3 bucket.

Below are a few commands that are super useful:

s3 make bucket

Quickly create a bucket from the command line. If you don’t specify –region the CLI will default to your default.

```
aws s3 mb s3://sqcows-bucket --region eu-west-1
```

s3 remove bucket

Use with care!!! The –force command is useful when you have a none empty bucket.

```
aws s3 rb s3://sqcows-bucket
```

```
aws s3 rb s3://sqcows-bucket --force # Delete a none empty bucket
```

s3 ls commands

Just like the *nix equivalent there is a recursive option which is useful when you start racking up lots of objects and prefixes.

```
aws s3 ls
```

```
aws s3 ls s3://sqcows-bucket
```

```
aws s3 ls s3://sqcows-bucket --recursive
```

```
aws s3 ls s3://sqcows-bucket --recursive  --human-readable --summarize
```

s3 cp commands

```
aws s3 cp getdata.php s3://sqcows-bucket
aws s3 cp /local/dir/data s3://sqcows-bucket --recursive

aws s3 cp s3://sqcows-bucket/getdata.php /local/dir/data

aws s3 cp s3://sqcows-bucket/ /local/dir/data --recursive

aws s3 cp s3://sqcows-bucket/init.xml s3://backup-bucket

aws s3 cp s3://sqcows-bucket s3://backup-bucket --recursive
```

s3 mv commands

```
aws s3 mv source.json s3://sqcows-bucket

aws s3 mv s3://sqcows-bucket/getdata.php /home/project

aws s3 mv s3://sqcows-bucket/source.json s3://backup-bucket

aws s3 mv /local/dir/data s3://sqcows-bucket/data --recursive

aws s3 mv s3://sqcows-bucket s3://backup-bucket --recursive
```

s3 rm commands

```
aws s3 rm s3://sqcows-bucket/<file_to_delete>

aws s3 rm s3://sqcows-bucket --recursive #delete all the files in the
bucket!!!!
```

s3 sync commands

I tend to use these over the cp commands as it works really well and feels more like rsync.

```
aws s3 sync backup s3://sqcows-bucket

aws s3 sync s3://sqcows-bucket/backup /tmp/backup

aws s3 sync s3://sqcows-bucket s3://backup-bucket
```

Infrastructure as Code

Automate all the things

OpEx Sec Rel Perf Cost Sus

The command line is great, but let's face it we are going to want to use the API more in our automation. When it comes to Infrastructure as code there as a few options and all have their own merits. Personally I tend to use terraform but there are other options like Amazons own Cloudformation or CDK.

https://learn.hashicorp.com/terraform

https://aws.amazon.com/cloudformation/

Why should you use IaC?

Having your infrastructure (in this case your S3 buckets) as code it not only helps you build your workload out but also allows you to replicate your set up in other environments and even allows you recovery more quickly in the event of a disaster situation, by rebuilding your setup quickly and reliability. By turning your infrastructure into deployable code you remove human error and lesson the changes of something working in stage and then failing in production.

Terraform

Terraform is an open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files.

Through out this book we’ll be using terraform, however you may already be an expert in cloudformation, so if you do feel inclined you could rewrite the examples, and we are always happy to accept pull requests to the book.

Tags

Tagging everything!

OpEx Sec Rel Perf Cost Sus

Why are tags important you may ask? Well, tags can help in a few ways such as cost allocation, defining the sensitivity of the data, what environment the data is for, and even access control. They can even be used to show the owner of certain files or workloads helping you quickly contact users, other companies use to tags to classify the sensitivity of the data in terms of Personal Identifiable Information (PII). Those PII tags then can be used to include/exclude access from IAM users.

Resource constraints

Each resource (a bucket in our case) can have up to 10 user-defined tags and these are broadly split into these areas, Technical, Automation, Business, and Security. Here are the restrictions you should consider when making tags:

Restriction	**Description**
Maximum number of tags per resource	50
Maximum key length	128 Unicode characters in UTF-8
Maximum value length	256 Unicode characters in UTF-8
Prefix restriction	Do not use the aws: prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.
Character restrictions	Tags may only contain Unicode letters, digits, whitespace, or these symbols: _ . : / = + - @

Recommended Tags

The table below shows the tags I recommend to customers, but of course, you can add more or leave some out.

Technical Tags	Automation Tags	Business Tags	Security Tags
Name: the application name	Created by: name the deployment system, CF, TF, or even by hand	Project: which project is the resource aligned too	Confidentiality: Mark if the data contains sensitive data such as personal information
Role: the application role/tier	Do Not Delete: true or false (allow the automation to check and bail out preventing data loss)	Owner: which team owns the data	Compliance: Does this data need to be handled in a certain way, such as HIPPA
Environment: dev/stage/prod etc	Date and Time: release date and time	Cost Centre: who should be billed for the usage	
Cluster: if the resource is used in a particular cluster tag it		Customer: Add this if you charge your customer for usage	
Version: stick to semver or git release if possible			

Using Terraform

Lets take a look at how this would look in terraform, the code can be found in the folder chapter1/001. It consists of five files:

main.tf

As the tiles suggest this is where we define the main stuff, as in what want to do. In this example, its create an S3 bucket and set versioning and tags on that bucket.

providers.tf

This is where we tell terraform to connect to and in our case it's AWS. We could also add some extra information to make sure the terraform state is stored remotely but that's beyond the scope of this book.

variables.tf

Anything we might want to change the value of in any of the other files should be stored as a variable in this file. You'll see them referenced in the other files like so: ${var.project} In this case this populates the project tag in the main.tf file.

versions.tf

We use this file to define the minimum version of the providers and modules we are using.

outputs.tf

The outputs file is a place where we can ask for information back from terraform, for instance, when a bucket is created.

To run this example make sure you are in the correct folder and run:

```
terraform init

terraform plan

terraform apply
```

Remember to answer yes when prompted on the apply. Once run you'll see a new bucket in your AWS console called sqcows-demo-bucket-

If you go and inspect that bucket you'll see versioning enabled and you can also inspect all the tags. The choice of tags that you require is up to you, however, one great way to find things that are not tagged in the correct way is to use AWS Config and use the required-tags rule to enforce all buckets be built with tags.

To clean up after this example just run the following command:

```
terraform destroy
```

Also type yes when prompted and it will clean up the deployment for you.

Technical considerations:

Which tags are useful tou your teams, having a owner and sensitivity level on data can help you prioritise work quickly in a crisis. You may also consider versioning is not required because you choose to replicate data to another region.

Business considerations:

What tags are important to the business, will they help you pin down the costs per click of operations for example? You should also be involved in setting the number of versions to keep. This could change from project to project but allows your business to know the risk and impact.

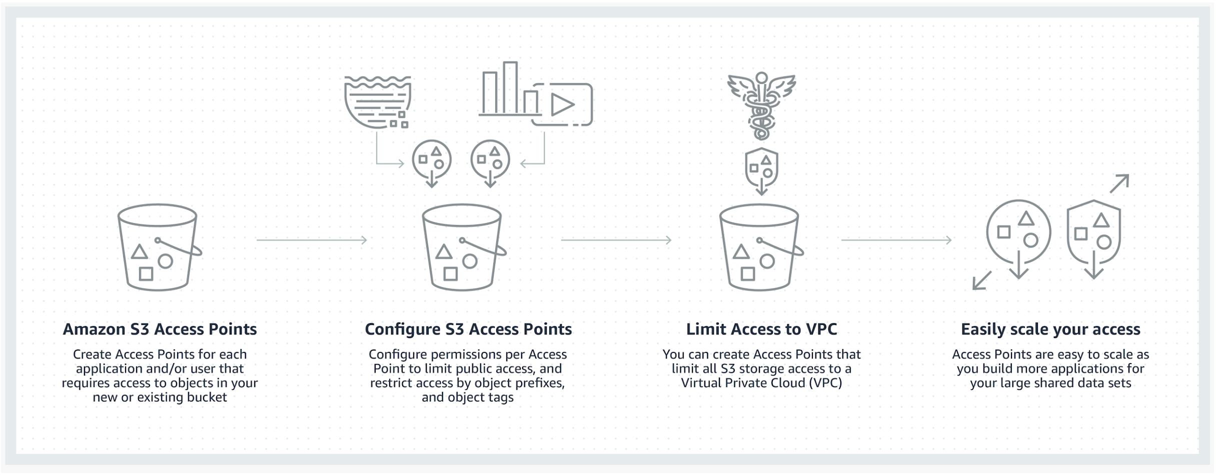
Access Points

Using access points to protect and control your data

How do S3 Access Points Work?

Access points are configured by policy to grant access to either specific users or applications. An example would be allowing groups of users (even from other accounts) access to your data lake.

Each access point is configured to connect to a single S3 bucket. The bucket then contains a network origin control (the source of where you'd like apps or users to connect from) and a Block Public Access Policy. You could, of course, use the CIDR block of your VPC as the access point or get more granular and only allow certain subnets or single IP's. You can even use this to grant certain origins access to objects with a certain prefix or specific tags.


Amazon S3 Access Points
Create Access Points for each application and/or user that requires access to objects in your new or existing bucket
Configure S3 Access Points
Configure permissions per Access Point to limit public access, and restrict access by object prefixes, and object tags
Limit Access to VPC
You can create Access Points that limit all S3 storage access to a Virtual Private Cloud (VPC)
Easily scale your access
Access Points are easy to scale as you build more applications for your large shared data sets

You can access the data in a shared bucket by either ARN directly or via its Alias when the operation requires a full bucket name.

When to use S3 Access Points

S3 Access Points simplify how you manage data access for your application set to your shared data sets on S3. You no longer have to manage a single, complex bucket policy with hundreds of different permission rules that need to be written, read, tracked, and audited. With S3 Access Points, you can now create application-specific access points permitting access to shared data sets with policies tailored to the specific application.

- Large shared data sets: Using Access Points, you can decompose one large bucket policy into separate, discrete access point policies for each application that needs to access the shared data set. This makes it simpler to focus on building the right access policy for an application, while not having to worry about disrupting what any other application is doing within the shared data set.

- Copy data securely: Copy data securely at high speeds between same-region Access Points using the S3 Copy API using AWS internal networks and VPCs. Restrict access to VPC: An S3 Access Point can limit all S3 storage access to happen from a Virtual Private Cloud (VPC). You can also create a Service Control Policy (SCP) and require that all access points be restricted to a Virtual Private Cloud (VPC), firewalling your data to within your private networks.
- Test new access policies: Using access points you can easily test new access control policies before migrating applications to the access point, or copying the policy to an existing access point.
- Limit access to specific account IDs: With S3 Access Points you can specify VPC Endpoint policies that permit access only to access points (and thus buckets) owned by specific account IDs. This simplifies the creation of access policies that permit access to buckets within the same account, while rejecting any other S3 access via the VPC Endpoint.
- Provide a unique name: S3 Access points allow you to specify any name that is unique within the account and region. For example, you can now have a “test” access point in every account and region.

Whether creating an access point for data ingestion, transformation, restricted read access, or unrestricted access, using S3 Access Points simplifies the work of creating, sharing, and maintaining access to data in your shared S3 buckets.

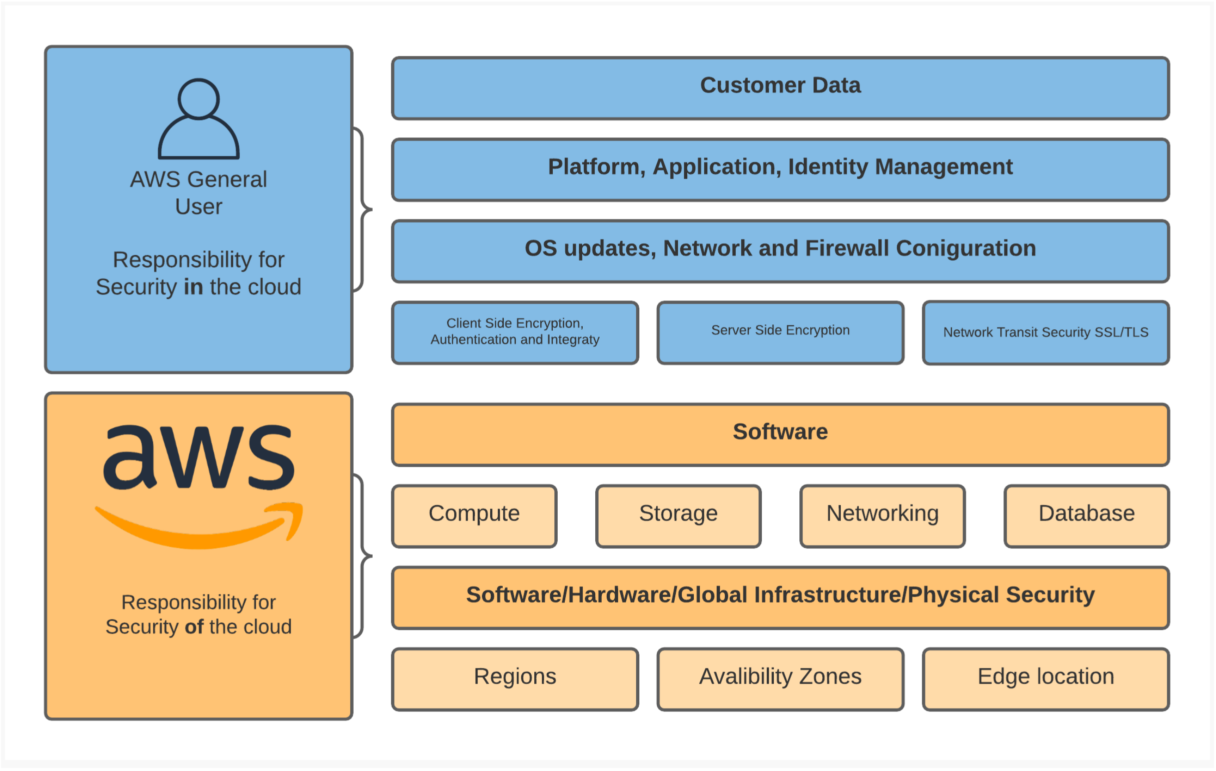
Security

Establish Best Practice to protect the confidentiality and integrity of data

Security speaks for itself and as Werner Vogals says “security is job zero”. In this pillar not only do we look at the data you are storing and encryption but we will consider labeling and identifying your data, marking it as sensitive if needed.

Shared Responsibility

Before we get going lets talk about shared responsibility. AWS is responsible for the running of the cloud, you are responsible for whats running in the cloud. If you think of this in terms on EC2, when you launch a server it’s running on hardware, amazon looks after that hardware, it’s connected to a network and amazon also look after that. Where you come is the instance it’s self that you launched, you’ll configure the security group around it, you also need to patch the software on it. It’s a pretty simple concept but one you need to understand. When it comes to S3 we have a similar story. S3 (the service) is run by amazon, this includes things like making sure all the disks are healthy and the API’s are up and running and a plethora of other things. When you create a bucket, by default it’s locked down theres no public access. In fact you have to work pretty hard to disable everything and expose your data these days! However, the data is your responsibility and this chapter has some reciepes in to help you look after that data that helps you security your data in the cloud! The diagram below shows the shared responsibility model and where you need to do your bit to protect your data.


AWS General User
Responsibility for Security **in** the cloud
Customer Data
Platform, Application, Identity Management
OS updates, Network and Firewall Coniguration
Client Side Encryption, Authentication and Integraty
Server Side Encryption
Network Transit Security SSL/TLS
aws
Responsibility for Security **of** the cloud
Software
Compute
Storage
Networking
Database
Software/Hardware/Global Infrastructure/Physical Security
Regions
Avalibility Zones
Edge location

Bucket Policies

Set out what you can and can't do within a bucket

OpEx Sec Rel Perf Cost Sus

Bucket Policies are applied by the account owner to the bucket. Only the account owner can do this. It uses the JSON format and has a limit of 20k in size. This policy is then used to grant access to your resources. Lets have a chat about some of the terminology used for Bucket Policies first by having a look at the following elements and what they mean and refer too:

Resources: These can be buckets, objects, access points, and jobs for which you can allow or deny permissions. You refer to them by Amazon Resource Name (ARN).

Actions: Each resource type supports fined grained control actions so you can give a user the least privilege they need. In the case of S3 it could like the following example:

```
"s3:CreateBucket",

"s3:ListAllMyBuckets",

"s3:GetBucketLocation"
```

Effect: An effect is applied to the user request as an action and are in the form of Allow and Deny. If you don’r explicitly grant an Allow the default action is deny.

Principal: The account or user who is allowed access to the actions and resources in the statement. In a bucket policy its the root user of the account.

Condition: Conditions apply to policies that are being acted on and can be use to further limit or increase the scope in that particular action. An example would be: `json "Condition": { "StringEquals": { "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID" }` There’s a great example for this in the form of you may wish to enable MultiFactorAuth when someone tries to delete an object. In the terms of a bucket policy you’d need something like the following policy:

```
{

  "Id": "Policy1640609878106",

  "Version": "2012-10-17",

  "Statement": [
    {
      "Sid": "Stmt1640609875204",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteBucketPolicy",
        "s3:DeleteObject",
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersion",
        "s3:DeleteObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:MultiFactorAuthAge": "true"
        }
      },
      "Principal": "*"
    }
  ]
}
```

Let’s take this example (chapter2/001) forward, by showing you how to add a policy. In this case, we are going to create an IAM role and then allow anyone with that role attached or anyone who can assume that role to List the S3 bucket contents. The code below enables this:

```
resource "aws_iam_role" "this" {
  assume_role_policy = <<EOF
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF
}

data "aws_iam_policy_document" "bucket_policy" {
  statement {
    principals {
      type        = "AWS"
      identifiers = [aws_iam_role.this.arn]
    }

    actions = [
      "s3:ListBucket",
    ]
    resources = [
      "arn:aws:s3:::${local.bucket_name}",
    ]
  }
}
```

Finally, let's make sure we attach the policy to the bucket. If you check the terraform you'll see two additional lines adding in the policy to the s3 bucket creation:

```
attach_policy = true
policy        = data.aws_iam_policy_document.bucket_policy.json
```

To run this example (chapter3/002) make sure you are in the correct folder and run:

```
terraform init
terraform plan
terraform apply
```

You'll need to type yes when prompted.

To cleanup run

```
terraform destroy
```

And answer yes to the prompt.

Technical considerations:

You need to consider permissions very seriously and work on the principle of least privilege, start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. You can even use the IAM Access Analyzer to monitor this.

Business Considerations:

Business leaders need to help identify whom within the company should and shouldn't hace access, they should also be considered in what data should be made public.

Bucket Permissions

Define security at the bucket level

OpEx Sec Rel Perf Cost Sus

You can also set bucket-level permissions in S3, which can override object-level permissions that someone tries to set. This is a super quick thing to enable in terraform and if you create a bucket in the AWS console it'll actually be defaulted to these values. Once again these things will project you from accidentally exposing data. All these tags below are pretty self-explanatory and included in the example.

```
# S3 bucket-level Public Access Block configuration

block_public_acls       = true

block_public_policy     = true

ignore_public_acls      = true

restrict_public_buckets = true
```

With policies you can set at a bucket level are as follows READ, WRITE, READ_ACP, WRITE_ACP and FULL_CONTROL. These can also be applied at an individual object level with the exception of WRITE which isn't applicable. Be particularly careful with the ACP rules as these can allow a user to set your access policy.

Take a look at the following to see how these permissions map to IAM permissions: https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html#permissions

Even if you enable all available ACL options in the Amazon S3 console, the ACL alone won't allow everyone to download objects from your bucket. However, depending on which option you select, any user could perform these actions:

- If you select List objects for the Everyone group, then anyone can get a list of objects that are in the bucket.
- If you select Write objects, then anyone can upload, overwrite, or delete objects that are in the bucket.
- If you select Read bucket permissions, then anyone can view the bucket's ACL.
- If you select Write bucket permissions, then anyone can change the bucket's ACL.

To prevent any accidental change to public access on a bucket's ACL, you can configure public access settings for the bucket. If you select Block new public ACLs and uploading public objects, then users can't add new public ACLs or upload public objects to the bucket. If you select Remove public access granted through public ACLs, then all existing or new public access granted by ACLs is respectively overridden or denied. The changes in our terraform set these permissions.

To run this example (chapter2/002) make sure you are in the correct folder and run:

```
terraform init
```

```
terraform plan
```

```
terraform apply
```

You'll need to type yes when prompted.

To cleanup run

```
terraform destroy
```

And answer yes to the prompt.

Technical considerations:

These policies can affect the way developers and applications can interact with the data stored in AWS. Some buckets are public for a reason, and having that knowledge and only allowing non-sensitive flagged objects in that bucket will help to keep you safe.

Business considerations:

Help the devop's teams identify data correctly. You may work with these files on a daily basis and are the best people to say this needs protecting at all costs or this data here is fine to be public. It's a team effort to secure the data.

Enabling Encryption

Types of encryption at rest

OpEx Sec Rel Perf Cost Sus

Just as Werner Vogals (CRO of AWS) said, dance like no one is watching, encrypt data like everyone is! So in this light, there are a few options. You could programmatically encrypt your data before you upload but this is going to require using your resources and managing your own key rotation. You can of course rely on AWS to make this easier for you. In (chapter3/004) we use code that lets AWS generate a key and manage it entirely for you. This is the default SSE-S3 method and it's best to enable it as a rule on the bucket, take note of the line:

```
server_side_encryption = "AES256"
```

This is the part that tells AWS to use default encryption. It's the cheapest and easiest way to encrypt at rest. You'll also see the code at the bottom that uploads our file. Note theres nothing special on that file to say "hey lets encrypt", it's automatically applied at the bucket level.

To run this example (chapter2/003) make sure you are in the correct folder and run:

```
terraform init

terraform plan

terraform apply
```

You'll need to type yes when prompted.

To cleanup run

```
terraform destroy
```

And answer yes to the prompt.

There is another way however and this is to use AWS Key management service (KMS) this is also referred to as SSE-KMS. This allows you to manage keys on a more granular level and rotate keys when and if needed. We will first need to define a KMS key:

```
resource "aws_kms_key" "objects" {
  description             = "KMS key is used to encrypt bucket objects"

  deletion_window_in_days = 7

}
```

The value deletion_window_in_days is how long the key is valid for. If you go over this time or delete the key you'll lose access to that data also!!!! To use SSE-KMS slightly tweak the rule in the s3 bucket:

```
kms_master_key_id = aws_kms_key.objects.arn

sse_algorithm     = "aws:kms"
```

Note that the object we upload has no changes because we are applying this at the bucket level.

To run this example (chapter3/005) make sure you are in the correct folder and run:

```
terraform init

terraform plan

terraform apply
```

You'll need to type yes when prompted.

To cleanup run

```
terraform destroy
```

And answer yes to the prompt.

Technical considerations

Remember if you use KMS, you must take responsibility for the management and access to that key. It's also going to greatly increase the number of API calls to KMS and this is going to bump your costs up. Encryption also always has a cost in terms of accessing the data, there's extra work to be done so expect that it'll be slightly slower to retrieve an object.

Business considerations

You need to be aware that SSE-KMS whilst being the form of encryption you have the most control over is going to increase your costs. However, it may be the more suitable encryption if the data has been classified as highly sensitive. You can use different encryption on different buckets, so classify your data accordingly and pick the right encryption for you, but please do encrypt!

Signed URLs

Share Files Securely

OpEx Sec Rel Perf Cost Sus

If all objects by default are private in a bucket you are going to need a mechanisim to share files to others at some point or another. Only the object owner has permission to access these objects. However, the object owner can optionally share objects with others by creating a presigned URL and setting an expiration time on how long that key is valid for.

One way to generate signed URL's is to run a script like the one below. Create a new file called signedurl.py and with your favourite editor add the following content:

```
#!/usr/bin/env python3


import boto3, argparse


parser = argparse.ArgumentParser()

parser.add_argument('-b','--bucket', help='Name of your S3 Bucket', required=True)

parser.add_argument('-o','--object', help='Name of the object + prefix in your bucket', required=True)

parser.add_argument('-t','--time', type=int, help='Expirery in seconds Default = 60', default=60)

args =  vars(parser.parse_args())




def get_signed_url(time, bucket, obj):

    s3 = boto3.client('s3')


    url = s3.generate_presigned_url('get_object', Params = { 'Bucket': bucket, 'Key': obj }, ExpiresIn = time)

    return url
```

```
try:

    url = get_signed_url(args['time'],args['bucket'], args['object'])

    print(url)

except:

    print('Something odd happened')
```

You'll need to set the permisions on the script so you can execute the file like so:

```
chmod +x signedurl.py
```

To execute the following code you need to supply **-b** which is the bucket name, **-o** which is the object and **-t** which is time in seconds. The example below grants access for 500 secons to a bucket named s3redirect and file called index.html

```
./signedurl.py -b s3redirect -o index.html -t 500
```

which would return you an answe like below:

```
https://s3redirect.s3.amazonaws.com/index.html?AWSAccessKeyId=AKIAVFTCNZXUBOD4
ZTX6&Signature=gptFG4W2hCoRCWmd%2FloDtvqUonc%3D&Expires=1640724520
```

You can of course can do this by using the CLI directly with the following commands:

```
aws s3 presign s3://sqcows-bucket/<file_to_share>

aws s3 presign s3://sqcows-bucket/<file_to_share> --expires-in 300 #5mins
```

Technical considerations

Whilst it is an easy way to share files to people who normally have ano access this could become a cumbersome process. It would make more sense to review the users access rights and grant them proper access to the files they need. This would also prevent the signed URL being used by others.

Business considerations

The signed URL can be used by anyone who recieves the link. This opens up substantial risks for a business in the case of a signed url being used for PII data and then that URL being leaked or eroniously sent to the wrong reciepiant on email.

MFA Delete

Prevent accidental deletion of objects

OpEx Sec Rel Perf Cost Sus

It is possible to prevent accidental deletion of objects in S3. However at the time of writing this doesn’t work through the mfa_delete terraform provider, so we are going to make the call direct to the API.

Once enabled users are rewquired to enter a MFA code when they try and delete objects, this can provide extra time to think before doing something that can break things.

```
aws --profile <my_profile> s3api put-bucket-versioning --bucket <bucket-name>
--versioning configuration 'MFADelete=Enabled,Status=Enabled' --mfa
'arn:aws:iam::<account-id>:mfa/root-account-mfa-device <mfacode>
```

Reliability

Distributed system design, recovery planning, and adapting to changing requirements

This pillar focuses on the business and technical aspects of your workload and data. What sort of recovery time is acceptable to the business if there is a failure? How long do you keep backups for? Do you need to go global or is a single AZ the right choice for you and the business? We'll also cover changes and incident response and recovery as well as change management.

Reliability in AWS regions/zones/s3

Letis first talk about how AWS is built. There are regions such as eu-west-1 and us-west-2. These are geographic locations for the clusters of data centres AWS opperate. They are also largely separated in most cases protecting them from a single region failure, it's not 100% fault tollerent but outages that affect all regions at once are a rare occassion because of region isolation. Within a region you have zones, for example *eu-west-1a*, *eu-west-1b* and *eu-west-1c*. These zones are huge, and not just a single data centre, they are clusters of datacentres that are hyper local to each other, however, each zone may be miles apart from another.

This gives you a massive amount of reliability if you run in all the zones in a region when coming to storage or compute. In Amazon S3 Standard your data is replicated in all three zones within a region and you are protected from a single zone failure. In fact untill you move into S3 One Zone Infrequent Access (S3 OneZone-IA) you are protected against failure. If you compare this to you datacentres you effectively have DR from the start with having to do anything!!!!

Some companies may want to go further though and this is where a skilled Solutions Architect will talk to the business and see if they want to go beyond the high avaliability in a region and set up replication to another region. The trade off here is cost and it will double your data store costs plus network transfer fees.

Versioning

Keep multiple versions of your objects

OpEx Sec Rel Perf Cost Sus

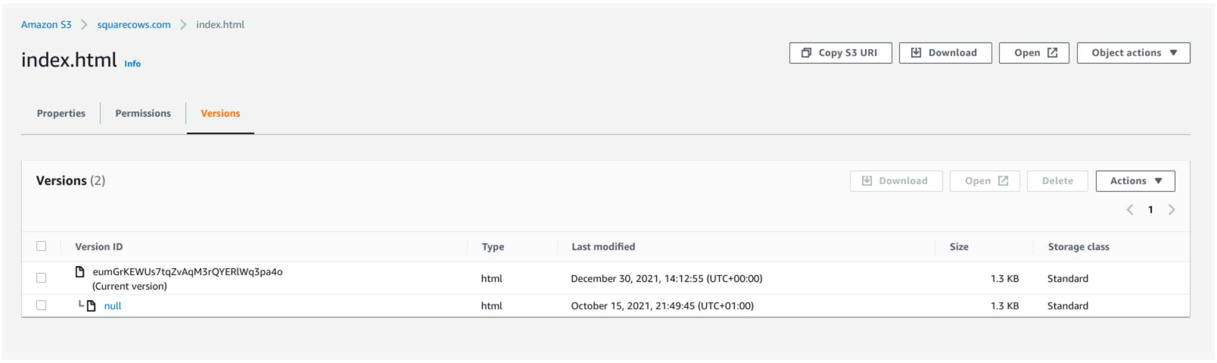
S3 allows you to keep multiple copies/variants of an object. This is really useful if you are updating a file and writing over the top but may need to revert to an older version for DR or other events where you may wish to restore a preserved object.

To enable versioning on S3 you do this at the bucket level and in terraform its super simple:

```
  versioning = {

      enabled = true

    }
```

Versioning even saves you from accidental deletion. If you delete an object when versioning is enabled, S3 flags the object with a delete marker instead of removing it permanently. By default, S3 Versioning is disabled on buckets, and you must explicitly enable it.

Unless you are using SDK's you are best using the AWS console, if you dive into the S3 console open a bucketed and look at an object thats recently been updated, you'll see the tags and how to restore to a previous version.



Versions (2)

Version ID	Type	Last modified	Size	Storage class
eumGrKEWUs7tqZvAqM3rQYERlWq3pa4o (Current version)	html	December 30, 2021, 14:12:55 (UTC+00:00)	1.3 KB	Standard
null	html	October 15, 2021, 21:49:45 (UTC+01:00)	1.3 KB	Standard

Technical considerations

Versioning is a great get out of jail free card should something (or someone) go wrong, it lets you got a stable earlier version. There is however a consideration on how long you keep the versions for and you'll need to discus this with the business.

Business considerations

If you keep multiple versions you end up with higher storage costs, so you’ll need to agree with the technical team if it’s needed or backups from another system would be better suited.

Same Region Replication

How S3 stores and replicates your data

OpEx Sec Rel Perf Cost Sus

Amazon S3 now supports automatic and asynchronous replication of newly uploaded S3 objects to a destination bucket in the same AWS Region. Amazon S3 Same-Region Replication (SRR) adds a new replication option to Amazon S3, building on S3 Cross-Region Replication (CRR) which replicates data across different AWS Regions.

Data stored in any Amazon S3 storage class, except for S3 One Zone-IA, is always stored across a minimum of three Availability Zones, each separated by miles within an AWS Region. SRR makes another copy of S3 objects within the same AWS Region, with the same redundancy as the destination storage class. This allows you to automatically aggregate logs from different S3 buckets for in-region processing, or configure live replication between test and development environments. SRR helps you address data sovereignty and compliance requirements by keeping a copy of your objects in the same AWS Region as the original.

When an S3 object is replicated using SRR, the metadata, Access Control Lists (ACL), and object tags associated with the object are also part of the replication. Once SRR is configured on a source bucket, any changes to the object, metadata, ACLs, or object tags trigger a new replication to the destination bucket.

In this example we are going to create a AWS bucket which will be a source and a destination bucket which is where we will replicate the too, the code can be found in the chapter3/001 folder. The example create both buckets and the replication policy,

```
resource "aws_iam_policy" "replication" {

  name        = var.iam_role_name

  description = "Replication Policy"

  policy      = <<EOF

{

    "Version": "2012-10-17",

    "Statement": [

        {

            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersionForReplication",
                "s3:GetObjectVersionAcl"
            ],
            "Resource": [
                "arn:aws:s3:::${aws_s3_bucket.source.id}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetReplicationConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::${aws_s3_bucket.source.id}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": "arn:aws:s3:::${aws_s3_bucket.destination.id}/*"
        }
    ]
}
EOF
}
```

This policy allows for replication to occur and in order for it to work we must attach it to a role:

```
resource "aws_iam_role" "replication" {
  name = var.iam_role_name
  tags = local.global_tags

  assume_role_policy = <<EOF
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
EOF
}
```

Finally we create a resource so we can use the ARN's in creating our source bucket. You'll see in the code that we have created a rule that copies everything in /copy_me to the destination bucket:

```
replication_configuration {

    role = aws_iam_role.replication.arn

    rules {

      id     = "destination"

      prefix = "DIRECTORY_NAME/"

      status = "Enabled"


      destination {

        bucket        = aws_s3_bucket.destination.arn

        storage_class = "STANDARD_IA"

      }

    }

  }
```

Once you've run terraform on this you simply need to drop a file in the copy_me folder and on the source bucket and then go have a look in the destination bucket.

Cross Region Replication

Replicate data to another region for backup

OpEx Sec Rel Perf Cost Sus

Amazon S3 cross region replication can be used for a few reasons. You may wish to have the data backed up 100's of miles away from your origin region for regulation reasons, you can also change acccount and ownership to prevent against accidental data loss. Another region maybe that you want to move data closer to the end user to reduce latancy. You can set cross region replication at a bucket level, a pre defined prefix or even on an object level with the correct tags.

There's a module for that

(https://github.com/asicsdigital/terraform-aws-s3-cross-account-replication)[https://github.com/asicsdigital/terraform-aws-s3-cross-account-replication]

Instead of reinventing the wheel it sometimes makes sense to use prebuilt modules that will get you up and running more quickly. This module helps you simply configure replication across regions and would be my recommended choice for the job in hand. It's supper simple and requires the following terraform variables:

Required

- source_bucket_name - Name for the source bucket (which will be created by this module)
- source_region - Region for source bucket
- dest_bucket_name - Name for the destination bucket (optionally created by this module)
- dest_region - Region for the destination bucket
- replication_name - Short name for this replication (used in IAM roles and source bucket configuration)
- aws.source - AWS provider alias for source account
- aws.dest - AWS provider alias for destination account

Optional

- create_dest_bucket - Boolean for whether this module should create the destination bucket
- replicate_prefix - Prefix to replicate, default "" for all objects. Note if specifying, must end in a /

Usage

```
provider "aws" {
  alias   = "source"
  profile = "source-account-aws-profile"
  region  = "us-west-1"
}

provider "aws" {
  alias   = "dest"
  profile = "dest-account-aws-profile"
  region  = "us-east-1"
}

module "s3-cross-account-replication" {
  source           =
"github.com/asicsdigital/terraform-aws-s3-cross-account-replication?ref=v1.0.0
"
  source_bucket_name = "source-bucket"
  source_region      = "us-west-1"
  dest_bucket_name   = "dest-bucket"
  dest_region        = "us-east-1"
  replication_name   = "my-replication-name"

  providers {
    "aws.source" = "aws.source"
    "aws.dest"   = "aws.dest"
  }
}
```

```
output "dest_account_id" {

  value = "${module.s3-cross-account-replication.dest_account_id}"

}
```

Amazon Replication Time Control

Amazon S3 replication time control helps you meet compliance “or business requirements” for data replication and provides visibility into Amazon S3 replication activity. Replication time control replicates most objects “that you upload” to Amazon S3 in seconds, and 99.99 percent of those objects within 15 minutes. S3 Replication Time Control, by default, includes S3 replication metrics and S3 event notifications, with which you can monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, and the maximum replication time.

Business considerations

Consider your users, if you have large data assets (video or audio, etc) then you’ll want this to be closer to the end user so they get a good experience. Even when using cloudfront, it can take time to come from eu-west-1 to ap-southeast-2 for example. So for smooth a smooth user experience consider this option. You may also have a regualtory reason to make sure backups exist 100’s of miles from the origin.

Performance Efficiency

Right size your resources and streamline you monitoring to deliver performant workloads

Performance covers a lot of ground and you really need to balance business and technical requirements here also. What’s an acceptable load time for your customers? If you get that wrong, customers can become frustrated and you’ll lose business. We could of course build something super responsive but waste resources and money, so there is a fine line to tread.

When we think about S3 and performance we no longer have to consider things like randomizing prefixes to spread the data out in order to avoid hot spots in data. Amazon listened to it’s customer and handled this for us. However there are things we can do still for performance, these can be from using batch to process lots of data using S3-Select to minimise the data you pullfrom s3 and that helps with cost also. But the easiest way to keep an eye on things is to use the S3 dashboard something that I geek out about on a regular basis because it has pretty graphs :) We’ll look at some of the operations in this chapter and I’ll show you how to keep an eye on systems and keep objects flying in and out of your buckets.

SNS and EventBridge

SNS and EventBridge Triggers.

OpEx Sec Rel Perf Cost Sus

Enabling notifications is a bucket-level operation. You store notification configuration information in the notification subresource that's associated with a bucket. After you create or change the bucket notification configuration, it usually takes about five minutes for the changes to take effect. When the notification is first enabled, an s3:TestEvent occurs. Amazon S3 stores the notification configuration as XML in the notification subresource that's associated with a bucket.

Technical Considerations

Using SNS or eventbridge can give you great flexibility to have actions performed when a file is upload/deleted/updated in S3. During this guide you'll also see you can additionally use SQS or trigger Lambda directly and you may wonder why not use these approaches instead, and you'd be right, It's more efficient to go direct to Lambda, however, SNS can deliver to multiple subscribers (lambda, email, etc) So it gives you a few more options. Eventbridge is also an enhancement over direct to Lambda as it allows you to filter which messages will actually triger Lambda running and potentially save you 1000's of unneeded invocations.

Business Considerations

Using cheaper storage such as S3 has a real potential to lower your bill, but you'll probably want to do something with that data. This chapter shows that S3 can be a power hub allowing your data to be automatically processed on update or other operations. This way of working can help your business transform to a micro-services style of working, which will help you gain speed in rolling out new features and updates without affecting the entire business, thus you can innovate faster.

Enabling EventBridge

EventBridge Triggers.

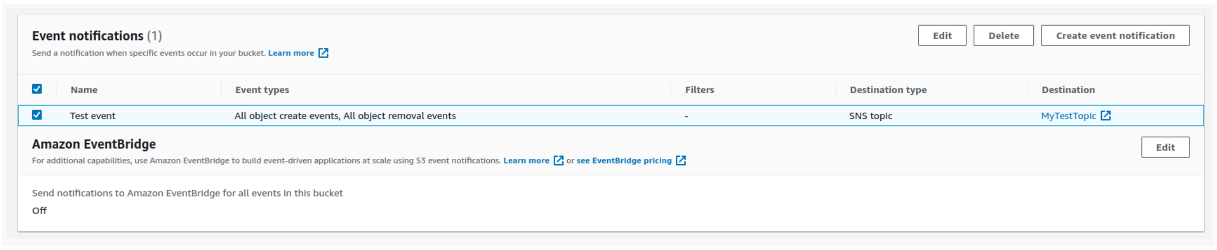
OpEx Sec Rel Perf Cost Sus

You can enable Amazon EventBridge using the S3 console, AWS Command Line Interface (AWS CLI), or Amazon S3 REST API.

Using the S3 console

To enable EventBridge event delivery in the S3 console.

- Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- In the Buckets list, choose the name of the bucket that you want to enable events for.
- Choose Properties.
- Navigate to the Event Notifications section and find the Amazon EventBridge subsection. Choose Edit.



Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

☑	Name	Event types	Filters	Destination type	Destination
☑	Test event	All object create events, All object removal events	-	SNS topic	MyTestTopic

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

Send notifications to Amazon EventBridge for all events in this bucket
Off

- Under Send notifications to Amazon EventBridge for all events in this bucket choose On.

Note After you enable EventBridge, it takes around five minutes for the changes to take effect.

Using the AWS CLI

The following example creates a bucket notification configuration for bucket with Amazon EventBridge enabled.

```
aws s3api put-bucket-notification-configuration --bucket <BUCKET-NAME>
--notification-configuration '{ "EventBridgeConfiguration": {} }'
```

Creating EventBridge rules

Once enabled you can create Amazon EventBridge rules for certain tasks. For example, you can send email notifications when an object is created.

SNS Topic Notifications

SNS Triggers.

OpEx Sec Rel Perf Cost Sus

Configuring event notifications via the console

Publish event messages to an SNS Topic

- Head to the SNS console and create a new topic, Just set the name and leave everything else as standard.
- Make a note of the ARN you'll need this in a second
- Now edit the SNS topic and edit the Acess Policy. We are going to narrow the policy down to SNS:Publish from your bucket only. Make sure your replace , and with your details:

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "<SNS-ARN>",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:<BUCKET-NAME>"
                },
                "StringEquals": {
                    "aws:SourceAccount": "<ACCOUNT-ID>"
                }
            }
        }
    ]
}
```

- Save your settings
- Now back on the S3 console select your bucket and click edit
- Click on the **Properties** tab and scroll down to **Notifications**

- Create a new notification and follow the settings in the following screen shot and be sure to select the correct SNS Topic!

Multipart Uploads

Speeding up uploads

OpEx Sec Rel Perf Cost Sus

If you are looking to upload an object greater than ~100mb in size you should consider using multipart uploads. This will speed up your total time to upload by using multiple threads. You also get the added bennefit that if a part fails in upload, you can just reupload that part and not the entire part again.

Using multipart upload provides the following advantages:

- Improved throughput - You can upload parts in parallel to improve throughput.
- Quick recovery from any network issues - Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- Pause and resume object uploads - You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or stop the multipart upload.
- Begin an upload before you know the final object size - You can upload an object as you are creating it.

There are three steps to a multipart upload,

- You initiate a request to S3 and get a UUID back
- You upload the parts
- You send a complete upload to S3 and S3 reconsructs the object for you

When uploading your file is split into parts, anywhere between w and 10,000, and if you are doing this programatically you need to track the part numbers and the ETag responses from S3 within your application.i The good news is the aws cli automatically splits large files for you. I've included information from the aws website below showing you how to upload and tweak how many concurrent connections you are making, so if you have a fast and stable connection you can use it to it's full potential, equally you could minimize the calls and slow down the amount of the concurrent uploads.

(Recommended) Upload the file using high-level (aws s3) commands

To use a high-level aws s3 command for your multipart upload, run this command:

```
$ aws s3 cp large_test_file s3://DOC-EXAMPLE-BUCKET/
```

This example uses the command aws s3 cp, but other aws s3 commands that involve uploading objects into an S3 bucket (for example, aws s3 sync or aws s3 mv) also automatically perform a multipart upload when the object is large.

Objects that are uploaded to Amazon S3 using multipart uploads have a different ETag format than objects that are uploaded using a traditional PUT request. To store the MD5 checksum value of the source file as a reference, upload the file with the checksum value as custom metadata. To add the MD5 checksum value as custom metadata, include the optional parameter –metadata md5="examplemd5value1234/4Q" in the upload command, similar to the following:

```
$ aws s3 cp large_test_file s3://DOC-EXAMPLE-BUCKET/ --metadata
md5="examplemd5value1234/4Q"
```

To use more of your host's bandwidth and resources during the upload, increase the maximum number of concurrent requests set in your AWS CLI configuration. By default, the AWS CLI uses 10 maximum concurrent requests. This command sets the maximum concurrent number of requests to 20:

```
$ aws configure set default.s3.max_concurrent_requests 20
```

Upload the file in multiple parts using low-level (aws s3api) commands

Important: Use this aws s3api procedure only when aws s3 commands don't support a specific upload need, such as when the multipart upload involves multiple servers, a multipart upload is being manually stopped and resumed, or when the aws s3 command doesn't support a required request parameter. For other multipart uploads, use aws s3 cp or other high-level s3 commands.

1. Split the file that you want to upload into multiple parts. Tip: If you're using a Linux operating system, use the split command.
2. Run this command to initiate a multipart upload and to retrieve the associated upload ID. The command returns a response that contains the UploadID:

```
aws s3api create-multipart-upload --bucket DOC-EXAMPLE-BUCKET --key
large_test_file
```

3. Copy the UploadID value as a reference for later steps.
4. Run this command to upload the first part of the file. Be sure to replace all values with the values for your bucket, file, and multipart upload. The command returns a response that contains an ETag value for the part of the

file that you uploaded. For more information on each parameter, see upload-part.

```
aws s3api upload-part --bucket DOC-EXAMPLE-BUCKET --key large_test_file
--part-number 1 --body large_test_file.001 --upload-id
exampleTUVGeKAk3Ob7qMynRKqe3ROcavPRwg92eA6JPD4ybIGRxJx9R0VbgkrnOVphZFK59KCYJAO
1PXlrBSW7vcH7ANHZwTTf0ovqe6XPYHwsSp7eTRnXB1qjx40Tk --content-md5
exampleaAmjr+4sRXUwf0w==
```

5. Copy the ETag value as a reference for later steps.
6. Repeat steps 4 and 5 for each part of the file. Be sure to increase the part number with each new part that you upload.
7. After you upload all the file parts, run this command to list the uploaded parts and confirm that the list is complete:

```
aws s3api list-parts --bucket DOC-EXAMPLE-BUCKET --key large_test_file
--upload-id
exampleTUVGeKAk3Ob7qMynRKqe3ROcavPRwg92eA6JPD4ybIGRxJx9R0VbgkrnOVphZFK59KCYJAO
1PXlrBSW7vcH7ANHZwTTf0ovqe6XPYHwsSp7eTRnXB1qjx40Tk
```

8. Compile the ETag values for each file part that you uploaded into a JSON-formatted file that is similar to the following:

```
{

    "Parts": [{

        "ETag": "example8be9a0268ebfb8b115d4c1fd3",

        "PartNumber":1

    },


    ....


    {

        "ETag": "example246e31ab807da6f62802c1ae8",

        "PartNumber":4

    }]

}
```

9. Name the file fileparts.json.
10. Run this command to complete the multipart upload. Replace the value for –multipart-upload with the path to the JSON-formatted file with ETags that you created.

```
aws s3api complete-multipart-upload --multipart-upload file://fileparts.json
--bucket DOC-EXAMPLE-BUCKET --key large_test_file --upload-id
exampleTUVGeKAk3Ob7qMynRKqe3ROcavPRwg92eA6JPD4ybIGRxJx9R0VbgkrnOVphZFK59KCYJAO
1PXlrBSW7vcH7ANHZwTTf0ovqe6XPYHwsSp7eTRnXB1qjx40Tk
```

11. If the previous command is successful, then you receive a response similar to the following:

```
{

    "ETag": "\\"exampleae01633ff0af167d925cad279-2\\"",

    "Bucket": "DOC-EXAMPLE-BUCKET",

    "Location": "https://DOC-EXAMPLE-BUCKET.s3.amazonaws.com/large_test_file",



    "Key": "large_test_file"

}
```

Resolve upload failures

If you use the high-level aws s3 commands for a multipart upload and the upload fails (due either to a timeout or a manual cancellation), you must start a new multipart upload. In most cases, the AWS CLI automatically cancels the multipart upload and then removes any multipart files that you created. This process can take several minutes.

If you use aws s3api commands for a multipart upload and the process is interrupted, you must remove incomplete parts of the upload, and then re-upload the parts.

To remove the incomplete parts, use the AbortIncompleteMultipartUpload lifecycle action. Or, use aws s3api commands to remove the incomplete parts by following these steps:

1. Run this command to list incomplete multipart file uploads. Replace the value for –bucket with the name of your bucket.

```
aws s3api list-multipart-uploads --bucket DOC-EXAMPLE-BUCKET
```

2. The command returns a message with any file parts that weren't processed, similar to the following:

```
{

    "Uploads": [

        {


    "Initiator": {

                "DisplayName": "multipartmessage",

                "ID":
"290xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

    "

            },

            "Initiated": "2016-03-31T06:13:15.000Z",


    "UploadId":
"examplevQpHp7eHc_J5s9U.kzM3GAHeOJh1P8wVTmRqEVojwiwu3wPX6fWYzADNtOHklJI6W6Q9NJ
UYgjePKCVpbl_rDP6mGIr2AQJNKB_A-",

            "StorageClass": "STANDARD",


    "Key": "",

            "Owner": {

                "DisplayName": "multipartmessage",


    "ID": "290xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx "

            }

        }

   ]
}
```

3. Run this command to remove the incomplete parts:

```
aws s3api abort-multipart-upload --bucket DOC-EXAMPLE-BUCKET --key
large_test_file --upload-id
examplevQpHp7eHc_J5s9U.kzM3GAHeOJh1P8wVTmRqEVojwiwu3wPX6fWYzADNtOHklJI6W6Q9NJU
YgjePKCVpbl_rDP6mGIr2AQJNKB
```

Batch

Performing Large Scale Batch opperations on S3

OpEx Sec Rel Perf Cost Sus

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can perform a single operation on lists of Amazon S3 objects that you specify. A single job can perform a specified operation on billions of objects containing exabytes of data. Amazon S3 tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, and serverless experience. You can use S3 Batch Operations through the AWS Management Console, AWS CLI, Amazon SDKs, or REST API.

Use S3 Batch Operations to copy objects and set object tags or access control lists (ACLs). You can also initiate object restores from S3 Glacier Flexible Retrieval or invoke an AWS Lambda function to perform custom actions using your objects. You can perform these operations on a custom list of objects, or you can use an Amazon S3 Inventory report to easily generate lists of objects. Amazon S3 Batch Operations use the same Amazon S3 APIs that you already use with Amazon S3, so you'll find the interface familiar.

S3 Batch Operations basics

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can run a single operation or action on lists of Amazon S3 objects that you specify.

Terminology

This section uses the terms jobs, operations, and tasks, which are defined as follows:

Job

A job is the basic unit of work for S3 Batch Operations. A job contains all of the information necessary to run the specified operation on the objects listed in the manifest. After you provide this information and request that the job begin, the job performs the operation for each object in the manifest.

Operation

The operation is the type of API action, such as copying objects, that you want the Batch Operations job to run. Each job performs a single type of operation across all objects that are specified in the manifest.

Task

A task is the unit of execution for a job. A task represents a single call to an Amazon S3 or AWS Lambda API operation to perform the job's operation on a single object. Over the course of a job's lifetime, S3 Batch Operations create one task for each object specified in the manifest.

How an S3 Batch Operations job works

A job is the basic unit of work for S3 Batch Operations. A job contains all of the information necessary to run the specified operation on a list of objects. To create a job, you give S3 Batch Operations a list of objects and specify the action to perform on those objects.

For information about the operations that S3 Batch Operations supports, see Operations supported by S3 Batch Operations.

A batch job performs a specified operation on every object that is included in its manifest. A manifest lists the objects that you want a batch job to process and it is stored as an object in a bucket. You can use a comma-separated values (CSV)-formatted Amazon S3 Inventory report as a manifest, which makes it easy to create large lists of objects located in a bucket. You can also specify a manifest in a simple CSV format that enables you to perform batch operations on a customized list of objects contained within a single bucket.

After you create a job, Amazon S3 processes the list of objects in the manifest and runs the specified operation against each object. While a job is running, you can monitor its progress programmatically or through the Amazon S3 console. You can also configure a job to generate a completion report when it finishes.

S3-Select

Filtering and retrieving data using Amazon S3 Select

OpEx Sec Rel Perf Cost Sus

With Amazon S3 Select, you can use simple structured query language (SQL) statements to filter the contents of an Amazon S3 object and retrieve just the subset of data that you need. By using Amazon S3 Select to filter this data, you can reduce the amount of data that Amazon S3 transfers, which reduces the cost and latency to retrieve this data.

Amazon S3 Select works on objects stored in CSV, JSON, or Apache Parquet format. It also works with objects that are compressed with GZIP or BZIP2 (for CSV and JSON objects only), and server-side encrypted objects. You can specify the format of the results as either CSV or JSON, and you can determine how the records in the result are delimited.

You pass SQL expressions to Amazon S3 in the request. Amazon S3 Select supports a subset of SQL. For more information about the SQL elements that are supported by Amazon S3 Select, see SQL reference for Amazon S3 Select and S3 Glacier Select.

You can perform SQL queries using AWS SDKs, the SELECT Object Content REST API, the AWS Command Line Interface (AWS CLI), or the Amazon S3 console. The Amazon S3 console limits the amount of data returned to 40 MB. To retrieve more data, use the AWS CLI or the API.

Requirements and limits

The following are requirements for using Amazon S3 Select:

- You must have s3:GetObject permission for the object you are querying.
- If the object you are querying is encrypted with a customer-provided encryption key (SSE-C), you must use https, and you must provide the encryption key in the request.

The following limits apply when using Amazon S3 Select:

- The maximum length of a SQL expression is 256 KB.
- The maximum length of a record in the input or result is 1 MB.
- Amazon S3 Select can only emit nested data using the JSON output format.
- You cannot specify the S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or REDUCED_REDUNDANCY storage classes. For more information, about storage classes see Storage Classes.

Additional limitations apply when using Amazon S3 Select with Parquet objects:

- Amazon S3 Select supports only columnar compression using GZIP or Snappy. Amazon S3 Select doesn't support whole-object compression for Parquet objects.
- Amazon S3 Select doesn't support Parquet output. You must specify the output format as CSV or JSON.
- The maximum uncompressed row group size is 512 MB.
- You must use the data types specified in the object's schema.
- Selecting on a repeated field returns only the last value.

Constructing a request

When you construct a request, you provide details of the object that is being queried using an InputSerialization object. You provide details of how the results are to be returned using an OutputSerialization object. You also include the SQL expression that Amazon S3 uses to filter the request.

For more information about constructing an Amazon S3 Select request, see SELECTObjectContent in the Amazon Simple Storage Service API Reference. You can also see one of the SDK code examples in the following sections.

Requests using scan ranges

With Amazon S3 Select, you can scan a subset of an object by specifying a range of bytes to query. This capability lets you parallelize scanning the whole object by splitting the work into separate Amazon S3 Select requests for a series of non-overlapping scan ranges. Scan ranges don't need to be aligned with record boundaries. An Amazon S3 Select scan range request runs across the byte range that you specify. A record that starts within the scan range specified but extends beyond the scan range will be processed by the query. For example; the following shows an Amazon S3 object containing a series of records in a line-delimited CSV format:

```
A,B

C,D

D,E

E,F

G,H

I,J
```

Use the Amazon S3 Select ScanRange parameter and Start at (Byte) 1 and End at (Byte) 4. So the scan range would start at "," and scan till the end of record starting at "C" and return the result C, D because that is the end of the record.

Amazon S3 Select scan range requests support Parquet, CSV (without quoted delimiters), and JSON objects (in LINES mode only). CSV and JSON objects must be uncompressed. For line-based CSV and JSON objects, when a scan range is specified as part of the Amazon S3 Select request, all records that start within the scan range are processed. For Parquet objects, all of the row groups that start within the scan range requested are processed.

Amazon S3 Select scan range requests are available to use on the Amazon S3 CLI, API and SDK. You can use the ScanRange parameter in the Amazon S3 Select request for this feature. For more information, see the Amazon S3 SELECT Object Content in the Amazon Simple Storage Service API Reference.

Errors

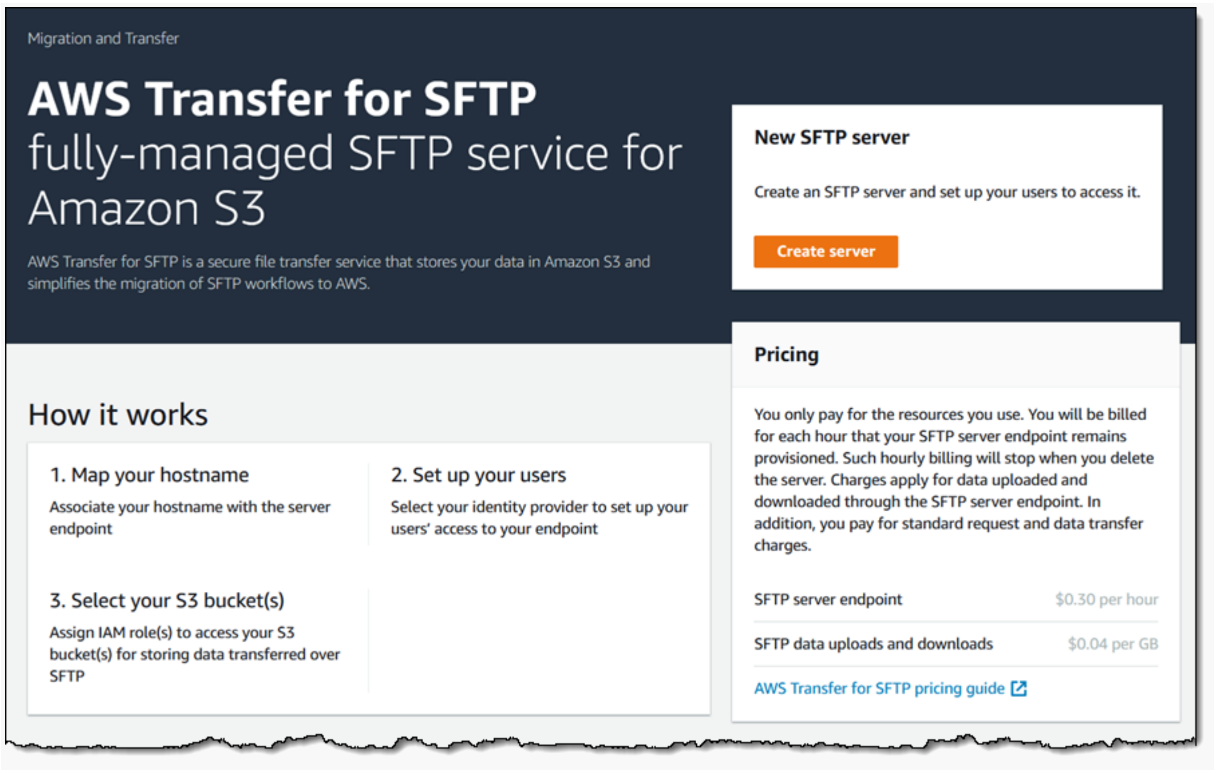
Amazon S3 Select returns an error code and associated error message when an issue is encountered while attempting to run a query. For a list of error codes and descriptions, see the List of SELECT Object Content Error Codes section of the Error Responses page in the Amazon Simple Storage Service API Reference.

SFTP Service

Allowing access to legacy applications

OpEx Sec Rel Perf Cost Sus

If you have applications or customers who need to transfer data in or out via SFTP, AWS Transfer for SFTP will help you. It allows clients to use the tools they are used to, but allows you take advantage of cheaper storage systems like S3! You can use the console to enable it but I've included source code to get this setup via terraform for you.



Migration and Transfer

AWS Transfer for SFTP
fully-managed SFTP service for Amazon S3

AWS Transfer for SFTP is a secure file transfer service that stores your data in Amazon S3 and simplifies the migration of SFTP workflows to AWS.

New SFTP server

Create an SFTP server and set up your users to access it.

How it works

1. Map your hostname
Associate your hostname with the server endpoint

2. Set up your users
Select your identity provider to set up your users' access to your endpoint

3. Select your S3 bucket(s)
Assign IAM role(s) to access your S3 bucket(s) for storing data transferred over SFTP

Pricing

You only pay for the resources you use. You will be billed for each hour that your SFTP server endpoint remains provisioned. Such hourly billing will stop when you delete the server. Charges apply for data uploaded and downloaded through the SFTP server endpoint. In addition, you pay for standard request and data transfer charges.

SFTP server endpoint	$0.30 per hour
SFTP data uploads and downloads	$0.04 per GB

AWS Transfer for SFTP pricing guide

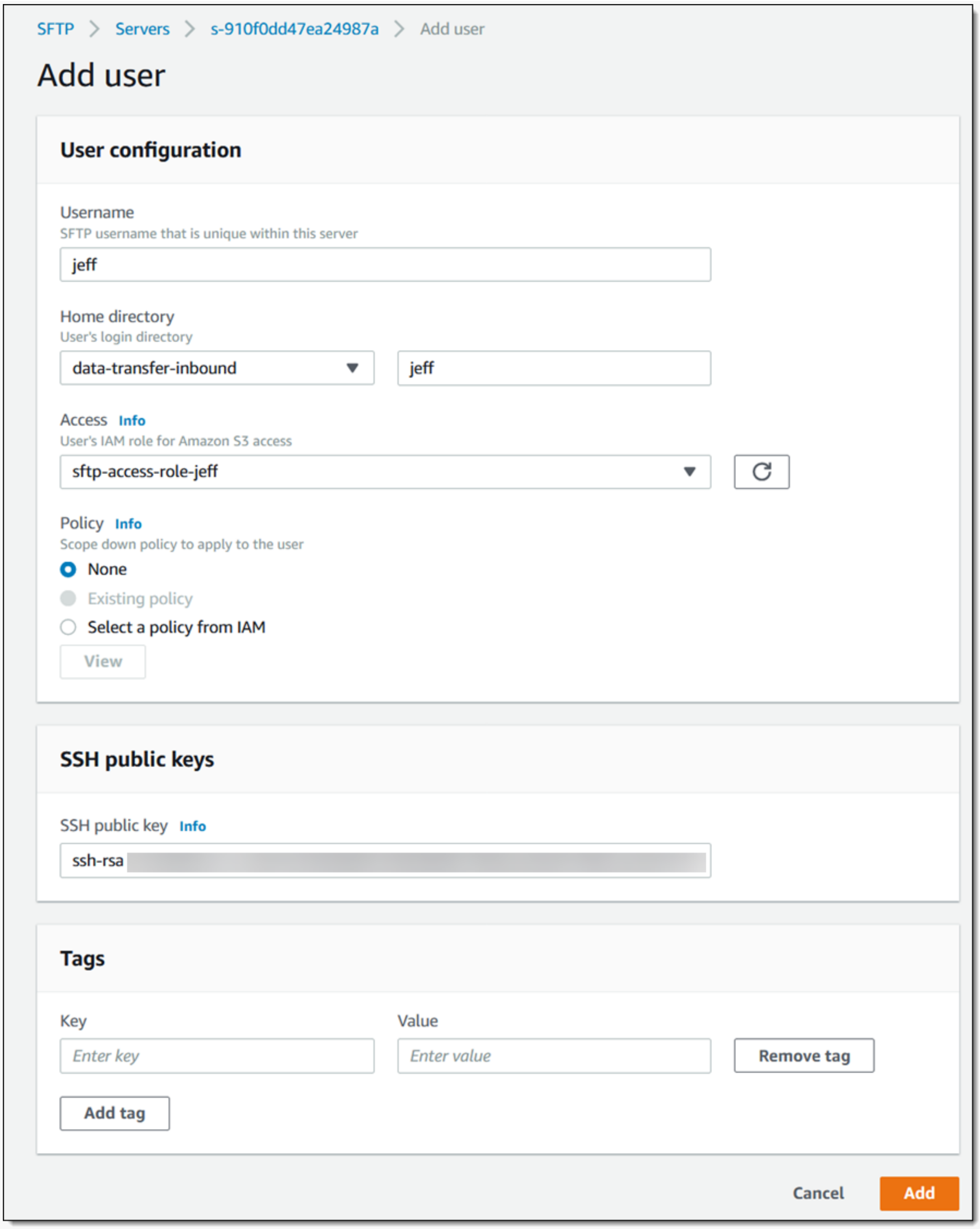
The code builds on our simple bucket example but adds in the transfer family:

```
resource "aws_transfer_server" "example" {

  security_policy_name = "TransferSecurityPolicy-2020-06"

  tags = {

      Name = local.bucket_name

      Project = "${var.project}"

      Environment = "${var.env}"

      Owner = "${var.owner}"
      CostCenter = "${var.cost}"

      Confidentiality = "${var.conf}"

  }

}
```

You can now take advantage of S3s scale and features such as versioning and intelligent tiering but access in a traditional way. The Final thing you'll need to do is add a user with a SSH key via the console. If you don't have a SSH already just run:

```
ssh-keygen
```

Now head to the AWS transfer pages in the Web Console for AWS. Here you will be able to add a user and your public key.


SFTP > Servers > s-910f0dd47ea24987a > Add user

Add user

User configuration

Username
SFTP username that is unique within this server
jeff

Home directory
User's login directory
data-transfer-inbound ▼ | jeff

Access Info
User's IAM role for Amazon S3 access
sftp-access-role-jeff ▼

Policy Info
Scope down policy to apply to the user
[✓ None]
Existing policy
Select a policy from IAM
View

SSH public keys

SSH public key Info
ssh-rsa [redacted]

Tags

Key	Value	
Enter key	Enter value	Remove tag

Add tag

Cancel Add

You can now connect either by the sftp command,

```
sftp localfile remote_file_directory/.
```

Or use a visual tool. You'll need the private key to connect.

Cost Optimization

The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point

The real art of any system is getting to grips with costs and maximizing the margins of your product. In terms of Amazon S3, this comes down to the storage medium you are using, you want to balance the best price with the performance and reliability pillars and it can be a tough line to walk.

If you are storing data on S3 rather than EBS (elastic block storage) volumes, EFS (elastic file storage), FSx for Lusture or even running proprietary cloud storage systems in the cloud, you are already doing well with cost optimization. By the way, Those proprietary systems are just using the same fundamental building blocks you have like EBS and adding a few features, seldom will they save you money. So the real trick to saving money in S3 is down to the data, you'll need to identify hot data that's always accessed and make sure that data that's cold gets moved to a cheaper tier. Also, this sounds simple if you don't need the data and the business agrees to delete it! I've seen customers with 8 years worth of snapshots for EBS and 10 years' worth of daily tgz files stored in an s3 bucket. To make matters worse there were several buckets identical which lead to approximately 100TB of storage that was never accessed, or likely to be! If you are wondering what that cost it was ~$24500.00 a month!!!! Yeah, I know!

Tiering

Understanding different tier cost and performance

OpEx Sec Rel Perf Cost Sus

Let's have a talk about the different storage tiers in S3. Amazon S3 offers a plethora of storage classes that you can choose from to best suit your needs around data access, resiliency, and cost requirements. Each storage class has it's merrits and in this section we'll look at which ones will be best for you. If you don't know which class of storage is right for your workload, or you have a workload with unpredictable patterns of usage you can even use S3 Intelligent-Tiering which automaticcally moves your data to the right class based on usage patterns, which will lean to lower costs for storaging your data. The there is **S3 Standard)** for frequently accessed data, the type of things you may access daily. This features

- Low latency and high throughput performance
- Designed for durability of 99.999999999% of objects across multiple Availability Zones (within a region)
- Resilient against events that impact an entire Availability Zone
- Designed for 99.99% availability over a given year
- Backed with the Amazon S3 Service Level Agreement for availability
- Supports SSL for data in transit and encryption of data at rest
- S3 Lifecycle management for automatic migration of objects to other S3 Storage Classes

The trade off here is that it costs more to have all those feature than other tiers. Saying that the cost isn't hugh for storage in S3 which is one of it's main attractions at *$0.023* per GB per month.

S3 Standard-Infrequent Access (S3 Standard-IA) at first glance looks almost identical to Standard S3. However it really is designed for files you don't open often. The costs is approx *$0.0.134* per GB per Month but if you switch this tier you may increate your API / object call pricing thus removing andy benefit.

- Same low latency and high throughput performance of S3 Standard
- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Resilient against events that impact an entire Availability Zone
- Data is resilient in the event of one entire Availability Zone destruction
- Designed for 99.9% availability over a given year
- Backed with the Amazon S3 Service Level Agreement for availability
- Supports SSL for data in transit and encryption of data at rest

- S3 Lifecycle management for automatic migration of objects to other S3 Storage Classes

S3 One Zone-Infrequent Access (S3 One Zone-IA) for less frequently accessed data. For me this is where it gets interesting. If you have data thats not accessed on a regular basis and can be recreated easily S3 One-Zone-IA could be for you. As you may have guessed instead of your data being replicated in all AZ's in a region this is going to live only in one. The Pay back for that is it's even cheeper at *$0.01048* per GB per Month. This risk to you though is that if an AZ has an issue you could loose the data.

- Same low latency and high throughput performance of S3 Standard
- Designed for durability of 99.999999999% of objects in a single Availability Zone†
- Designed for 99.5% availability over a given year
- Backed with the Amazon S3 Service Level Agreement for availability
- Supports SSL for data in transit and encryption of data at rest
- S3 Lifecycle management for automatic migration of objects to other S3 Storage Classes

Now lets take a look at the **Glacier** side of things. Glacier is cold storage designed for archiving data that you may never touch again or only in very certain situations. Think of a audit happening at work and they need records from 6 years ago. It would be expensive to store 6 years worth of data in standard S3 so this is where Glacier comes in.

Released in 2021 **S3 Glacier Instant Retrieval** is for archive data that needs immediate access. With Glacier you normally have a time delay to get the data back you are archiving, anywhere between 1-12 hours but Instant retrieval does what it says on the tin and you can **instantly get your databack**. As long as this is the sort of data you only access once a quarter it could be a good place for you to store data that fits this requirement. With a cost *$0.005* per GB per hour for data you may need to access its a good place to store it.

- Data retrieval in milliseconds with the same performance as S3 Standard
- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Data is resilient in the event of the destruction of one entire Availability Zone
- Designed for 99.9% data availability in a given year
- 128 KB minimum object size
- Backed with the Amazon S3 Service Level Agreement for availability
- S3 PUT API for direct uploads to S3 Glacier Instant Retrieval, and S3 Lifecycle management for automatic migration of objects

S3 Glacier Flexible Retrieval (formerly S3 Glacier) for rarely accessed long-term data that does not require immediate access is yet another tier of storage, With a slightly lower cost of *$0.00408* per GB per Month. It doesn't sound like a big reduction but when you are storing petabytes of data it can really add up quickly. The downside is that you need to put in a request for this data to be retrieved and it can take **1-12 hours** to do so.

- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Data is resilient in the event of one entire Availability Zone destruction
- Supports SSL for data in transit and encryption of data at rest
- Ideal for backup and disaster recovery use cases when large sets of data occasionally need to be retrieved in minutes, without concern for costs
- Configurable retrieval times, from minutes to hours, with free bulk retrievals
- S3 PUT API for direct uploads to S3 Glacier Flexible Retrieval, and S3 Lifecycle management for automatic migration of objects

Finally **Amazon S3 Glacier Deep Archive (S3 Glacier Deep Archive)** for long-term archive and digital preservation with retrieval in hours at the lowest cost storage in AWS's object storage tiers at *$0.0018* per GB per Month, the downside is it takes **12 hours** to restore your objects back into S3.

- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Lowest cost storage class designed for long-term retention of data that will be retained for 7-10 years
- Ideal alternative to magnetic tape libraries
- Retrieval time within 12 hours
- S3 PUT API for direct uploads to S3 Glacier Deep Archive, and S3 Lifecycle management for automatic migration of objects

There is technically another class of S3 storage that lives on AWS Outposts, which are appliances that you can store in your own DC but this book isn't going to cover that but can get more information here: https://aws.amazon.com/outposts/

Intelligent Tiering

Using the best price storage with intelligent tiering

OpEx Sec Rel Perf Cost Sus

When I tell you intelligent tiering is going to automatically save you money you might seem a little skeptical, but it really can in the right hands. The thing is there is a catch, but it’s not a big one. That is that aws charge you a small fee, but it really is small unless you have billions of objects in S3 as it’s about $0.10 per million objects monitored a month.

S3 Intelligent-Tiering is the ideal storage class for data with unknown, changing, or unpredictable access patterns, independent of object size or retention period. You can use S3 Intelligent-Tiering as the default storage class for virtually any workload, especially data lakes, data analytics, new applications, and user-generated content. The only caviet is that it doesn’t include objects under 128 KB, however they also don’t count in the charges for automation. These small objects remain in the frequent access tier.

How it works

Intelligent tiering is designed to optimise where you data is stored based on access patterns from daily use. It will then move lesser used objects onto cheaper tiers of S3 storage, when those patterns of access change it can also move the data back into the higher tiers. It’s configured with three tiers, *frequent* use, *infrequent* use and a very low cost tier for data *rarely* accessed. After 30 days of an not being access ed Intelligent tiering automatically moves the data to the lower cost storage S3 IA which should save you about 40% on your storage costs. After 90 days of no access Inteligent Tiering will move the data to Glacier Instant Access which brings your savings to 68% over standard storage costs. You can aditionally configure and opt in to Glacier Deep Archive which would boost your savings to 95%.

Life Cycle Policies

Save money on storage

OpEx Sec Rel Perf Cost Sus

In order to maximise your savings when using object storage you have to use the right tier for that data. You will need to balance cost against speed, reliability and avaliability. AWS Glacier Deep Archive is the cheapest storage but is a 12 hour wait to get your data back going to be a anacceptable buisiness requirement. When you decide the right balance in order to move the data between tiers you can imppliment Life Cycle policies. An S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:

Transition actions – These actions define when objects transition to another storage class. For example, you might choose to transition objects to the S3 Standard-IA storage class 30 days after creating them, or archive objects to the S3 Glacier Flexible Retrieval storage class one year after creating them.

There are costs associated with lifecycle transition requests and moving between tiers, the costs are minimal and however the savings of using a cheaper storage should be greater.

Expiration actions – These actions define when objects expire. Amazon S3 deletes expired objects on your behalf.

Managing object lifecycle

Define S3 Lifecycle configuration rules for objects that have a well-defined lifecycle. For example:

- If you upload periodic logs to a bucket, your application might need them for a week or a month. After that, you might want to delete them.
- Some documents are frequently accessed for a limited period of time. After that, they are infrequently accessed. At some point, you might not need real-time access to them, but your organization or regulations might require you to archive them for a specific period. After that, you can delete them.
- You might upload some types of data to Amazon S3 primarily for archival purposes. For example, you might archive digital media, financial and healthcare records, raw genomics sequence data, long-term database backups, and data that must be retained for regulatory compliance.

With S3 Lifecycle configuration rules, you can tell Amazon S3 to transition objects to less-expensive storage classes, or archive or delete them.

Using Terraform to manage your lifecycle

If we take a look at example code we have been working on we can add the required section in there.

```
 lifecycle_rule = [
    {
      id      = "log"
      enabled = true
      prefix  = "log/"

      tags = {
        rule      = "log"
        autoclean = "true"
      }

      transition = [
        {
          days          = 30
          storage_class = "ONEZONE_IA"
          }, {
          days          = 60
          storage_class = "GLACIER"
        }
      ]

      expiration = {
        days = 90
      }
      noncurrent_version_expiration = {
        days = 30
      }
    },
    {
      id                                     = "log1"
      enabled                                = true
      prefix                                 = "log1/"
      abort_incomplete_multipart_upload_days = 7

      noncurrent_version_transition = [
        {
          days          = 30
          storage_class = "STANDARD_IA"
        },
        {
          days          = 60
          storage_class = "ONEZONE_IA"
        },
        {
          days          = 90
          storage_class = "GLACIER"
        },
      ]

      noncurrent_version_expiration = {
        days = 300

      }

    },

  ]
```

The rule above has multiple parts. Anything in the log prefix (folder) will move to One Zone IA, After 60 days the data will move to Glacier and after 90 days the data will be deleted. Theres also a second rule for the profix log1, it'll move data to Standard IA after 30 days and then into One Zone IA after 60 days, after 90 days they'll move to Glacier and then finially 300 days later the data will be deleted. You will have noticed that the first rule is applied in a different way to the other. The second rule is used to clean up none current versions of files. As files change you can end up with a big long list of old versions. So the second rule is very importnat.

Technical considerations

Rememeber to clean up the versions, they're not visable but can add a sizable cost to the monthly bill if not managed. If you are underr GDPR you need to work out a way of expunging a persons data in these versions and backups also.

Business considerations

The business should supply the information for retention policies. Theres often rule and regulation around how long you must store data for.

S3 Lens

Monitor and act on your usage

OpEx Sec Rel Perf Cost Sus

Amazon S3 Storage Lens provides a single pane of glass approach to your object usage and activity across your entire Amazon S3 storage, either in a single account or across an organization.

You can collect activity metrics together and display them in a dashboard. You can even download the data in CSV or Parquet format.

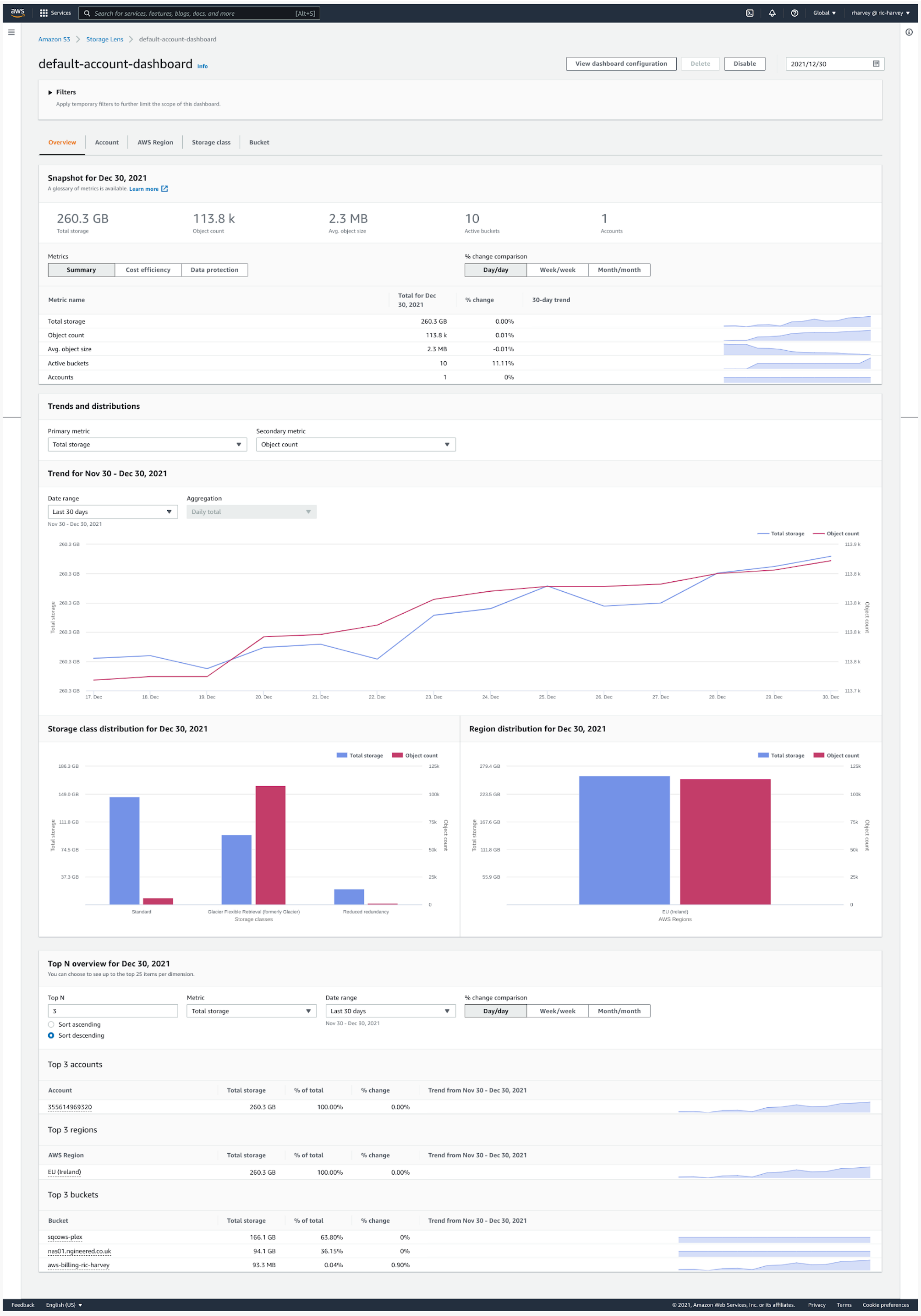
Configuration

Amazon S3 Storage Lens requires a configuration that contains the properties that are used to aggregate metrics on your behalf for a single dashboard or export. This includes all or partial sections of your organization account's storage, including filtering by Region, bucket, and prefix-level (available only with advanced metrics) scope. It includes information about whether you chose free metrics or advanced metrics and recommendations. It also includes whether a metrics export is required, and information about where to place the metrics export if applicable.

Default dashboard

The S3 Storage Lens default dashboard on the console is named default-account-dashboard. S3 preconfigures this dashboard to visualize the summarized insights and trends of your entire account's aggregated storage usage and activity metrics, and updates them daily in the Amazon S3 console. You can't modify the configuration scope of the default dashboard, but you can upgrade the metrics selection from free metrics to the paid advanced metrics and recommendations. You can also configure the optional metrics export, or even disable the dashboard. However, you can't delete the default dashboard.

The screenshoot below shows a sample of the default dashboard and if you are a graph nerd like myself or good friend Kieren you are in for a treat:


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
rharvey @ ric-harvey
Amazon S3 > Storage Lens > default-account-dashboard

default-account-dashboard Info

View dashboard configuration
Delete
Disable

2021/12/30

Filters

Apply temporary filters to further limit the scope of this dashboard.

Overview
Account
AWS Region
Storage class
Bucket

Snapshot for Dec 30, 2021

A glossary of metrics is available. Learn more

260.3 GB	113.8 k	2.3 MB	10	1
Total storage	Object count	Avg. object size	Active buckets	Accounts

Metrics: [✓ Summary] Cost efficiency Data protection

% change comparison: [✓ Day/day] Week/week Month/month

Metric name	Total for Dec 30, 2021	% change	30-day trend
Total storage	260.3 GB	0.00%	
Object count	113.8 k	0.01%	
Avg. object size	2.3 MB	-0.01%	
Active buckets	10	11.11%	
Accounts	1	0%	

Trends and distributions

Primary metric: Total storage

Secondary metric: Object count

Trend for Nov 30 - Dec 30, 2021

Date range: Last 30 days

Aggregation: Daily total

Nov 30 - Dec 30, 2021

Storage class distribution for Dec 30, 2021

Region distribution for Dec 30, 2021

Top N overview for Dec 30, 2021

You can choose to see up to the top 25 items per dimension.

Top N: 3

Sort ascending [✓ Sort descending]

Metric: Total storage

Date range: Last 30 days

Nov 30 - Dec 30, 2021

% change comparison: [✓ Day/day] Week/week Month/month

Top 3 accounts

Account	Total storage	% of total	% change	Trend from Nov 30 - Dec 30, 2021
355614969320	260.3 GB	100.00%	0.00%	

Top 3 regions

AWS Region	Total storage	% of total	% change	Trend from Nov 30 - Dec 30, 2021
EU (Ireland)	260.3 GB	100.00%	0.00%	

Top 3 buckets

Bucket	Total storage	% of total	% change	Trend from Nov 30 - Dec 30, 2021
sqcows-plex	166.1 GB	63.80%	0%	
nas01.ngineered.co.uk	94.1 GB	36.15%	0%	
aws-billing-ric-harvey	93.3 MB	0.04%	0.90%	

Sustainability

The Sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage.

Sustainability is a new pillar introduced in December 2021 and asks us as developers/engineers/architects/operations to think about the resources we are using, For example, do you need to run a server that offers uploads 24x7 when uploads only happen in business hours, if you don’t need it, automate its shutdown and start-up. You’ll also find that this has a pleasant effect on the cost optimization pillar due to the pay-for-what-you-use model of AWS.

Static Web Hosting

How to host simple websites

OpEx Sec Rel Perf Cost Sus

Static web hosting is one of my favourite parts of S3. You can simply upload a static site to a bucket and serve the pages straight away. It's worth noting S3 doesn't support any serverside processing, so you need a static website. There are lots of options here, my favourite is goHugo but I've included a list for you to get started:

- https://gohugo.io/
- https://www.gatsbyjs.com/
- https://jekyllrb.com
- https://nextjs.org
- https://react-static.js.org/
- https://vuepress.vuejs.org

Serving a static website from a traditional server is wasteful. It sits idel most of the time, you may even runn XX more servers ready for peak load wasting even more enegery. S3 web hosting is different. The webserveres only lift your content into memory to serve when a request comes in. AT times when there is no traffic you data storage is all that is taking energy thus greatly reducing your usage and my reasoning for making this section a sustainability section. It's also increadly cheep to host your site in this way.

By default, you'll be serving your website from its bucket name and region name. For example:

- s3-website dash (-) Region - http://bucket-name.s3-website-Region.amazonaws.com

It's also worth noting that the endpoint is http and not https!

Adding a DNS CNAME

If you have a registered domain, you can add a DNS **CNAME** entry to point to the Amazon S3 website endpoint. For example, if you registered the domain www.squarecows.com, you could create a bucket called www.squarecows.com, and add a DNS CNAME record that points to www.squarecows.com.s3-website.Region.amazonaws.com. All requests to http://www.squarecows.com are routed to www.squarecows.com.s3-website.Region.amazonaws.com.

Whilst this is a nicer way to host your site with a friendly URL it still lacks HTTPS and these days that will affect your search engine rankings.

The Apex zone issue!

Whilst Amazon Route53 allows you to point the apex record (in this case squarecows.com) at a bucket by a CNAME or alias, manny other providers will not allow you to do this because it's against the RFC. Instead you are required to point at an IP address. Theres no real fix for this and I would recomend moving you DNS hosting into Route53 to avoid potential issues.

Enabling Hosting

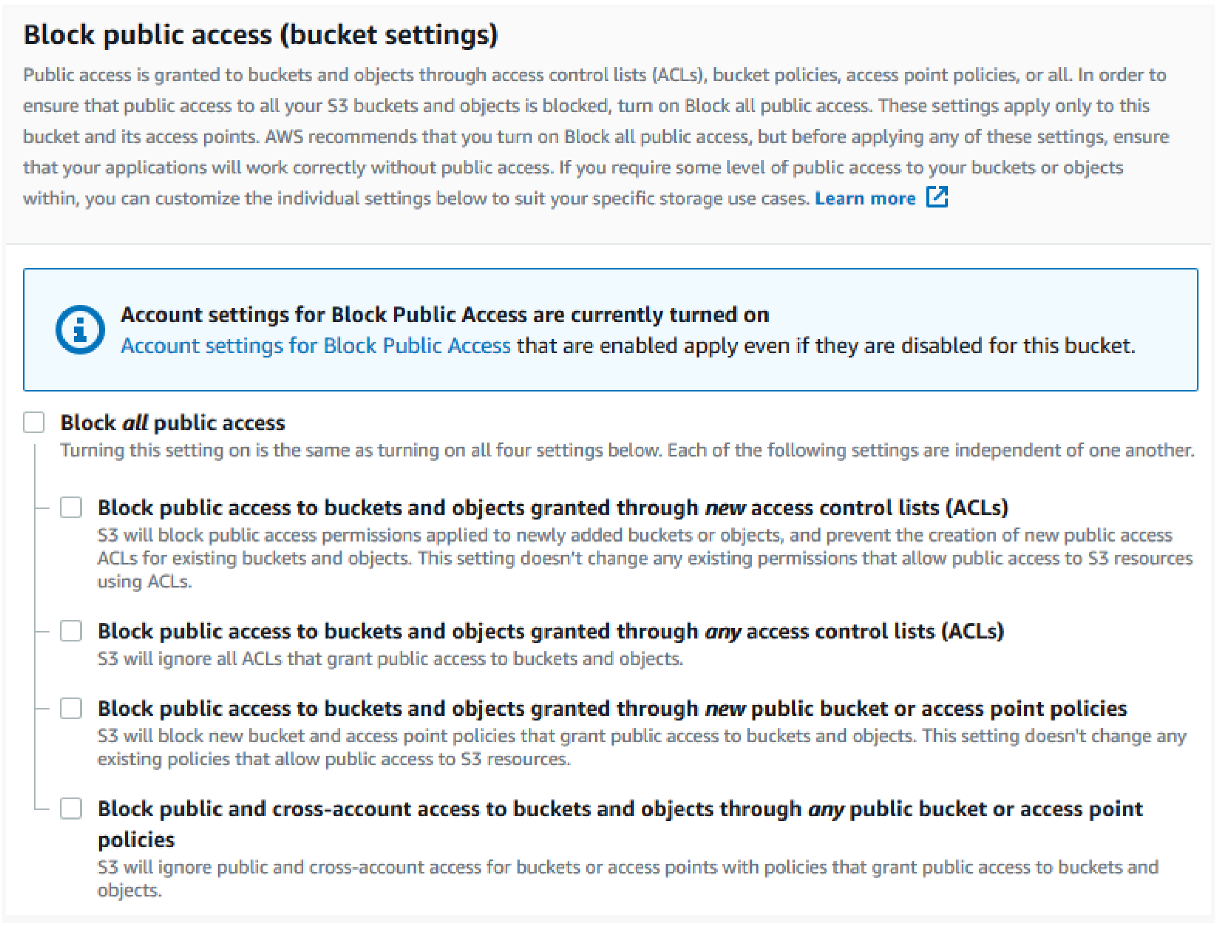
I've included the terraform to help you create an S3 website, however, you can also do this from the AWS CLI. When you create the website endpoint you need to specify your index page (normally an object named *index.html*) and your error document (normally and object nammed *error.html*).

```
aws s3 website s3://my-bucket/ --index-document index.html --error-document
error.html
```

For the Terraform check out FHGJDHFSFSGDSHJGHFJ

You are also required to make sure the bucket is publically readable so you need to grant access as *public* and remove the disable public block section from your bucket configuration. If you have objects that are uploaded by people other than the bucket owner you will also have to grant read access to everybody. Here are the steps to disable the public block which is on by default:

- Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- Choose the name of the bucket that you have configured as a static website.
- Choose Permissions.
- Under Block public access (bucket settings), choose Edit.
- Clear Block all public access, and choose Save changes.


Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

Account settings for Block Public Access are currently turned on
Account settings for Block Public Access that are enabled apply even if they are disabled for this bucket.

☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Now you've cleard these block policies you must do the bit I always for get to do! Add a bucket policiy to allow all the objects to be readable by everyone! This basically ensures all objects have the s3:GetObject permission.

- Under Buckets, choose the name of your bucket.
- Choose Permissions.
- Under Bucket Policy, choose Edit.
- To grant public read access for your website, copy the following bucket policy, and paste it in the Bucket policy editor.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

- Update the Resource to your bucket name. In the preceding example bucket policy, Bucket-Name is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.
- Choose Save changes. A message appears indicating that the bucket policy has been successfully added.
-

Execute code from S3

Execute code using S3 triggers

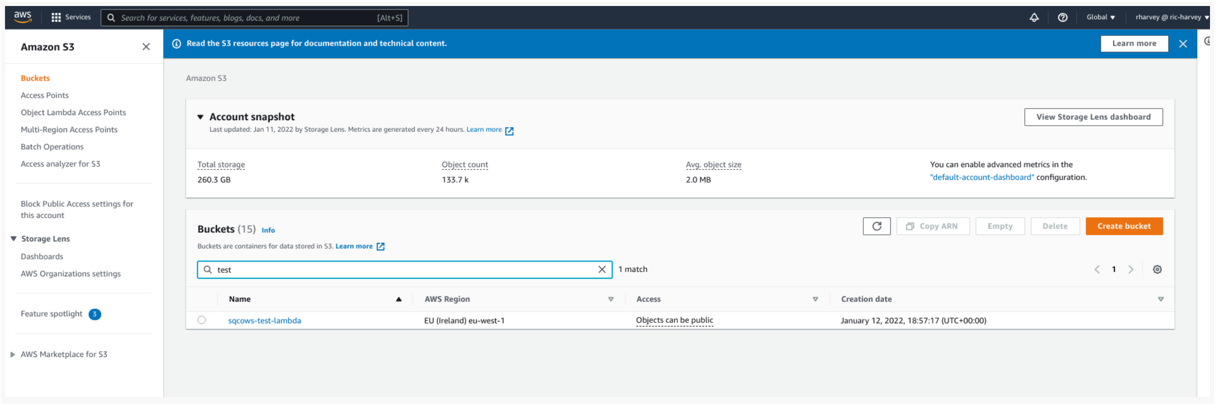
OpEx Sec Rel Perf Cost Sus

One of the biggest announcements to ever come out of re:invent (the AWS annual event in Las Vegas) was the launch of Lambda. Lambda is effectively Functions as a service. You can write code and give it to AWS and they handle the runtime and scaling of the function. Lambda supports multiple languages, we will be using python in this example. Lambda supports multiple triggers. Triggers are what tells Lambda to run the code you've written. You could use SNS or Event bridge as mentioned in previous chapters, however, Lambda natively supports a direct trigger from S3 on completion of the upload of an object. This is my mind is the jewl that makes S3 so awesome.

Create a Bucket

First, let's create a new bucket to test our trigger. From the command line run:

```
aws s3 mb s3://<MY_BUCKET_NAME>
```


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
rharvey @ ric-harvey
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
AWS Marketplace for S3
Learn more
View Storage Lens dashboard
Copy ARN
Empty
Delete
Create bucket

Read the S3 resources page for documentation and technical content.

Amazon S3

Account snapshot

Last updated: Jan 11, 2022 by Storage Lens. Metrics are generated every 24 hours. Learn more

Total storage	Object count	Avg. object size	
260.3 GB	133.7 k	2.0 MB	You can enable advanced metrics in the "default-account-dashboard" configuration.

Buckets (15) Info

Buckets are containers for data stored in S3. Learn more

test — 1 match

Name	AWS Region	Access	Creation date
sqcows-test-lambda	EU (Ireland) eu-west-1	Objects can be public	January 12, 2022, 18:57:17 (UTC+00:00)

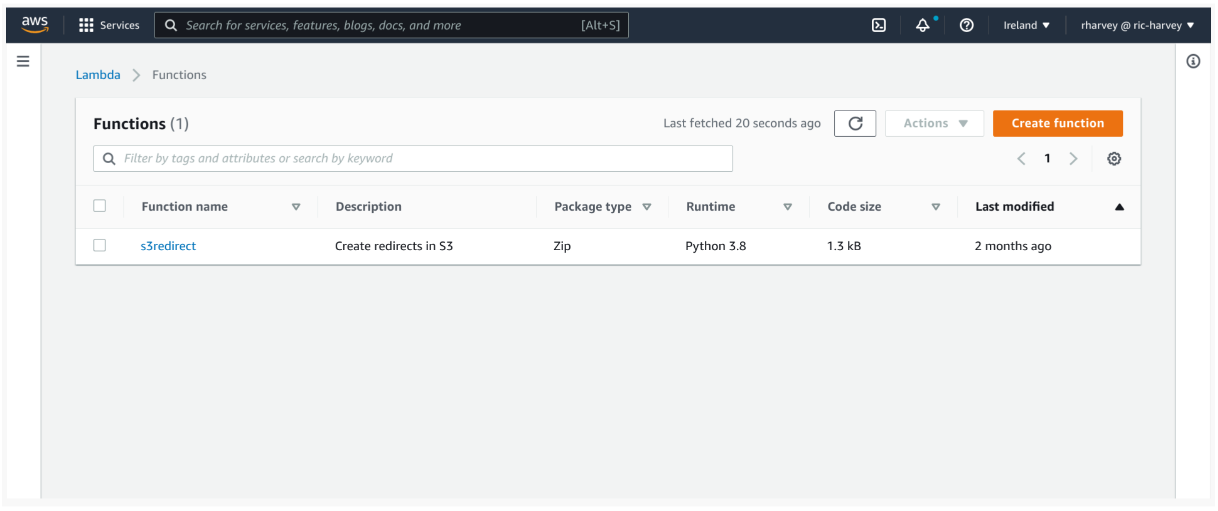
If you want to check fromt he CLI run:

```
aws s3 ls
```

Create the Lambda Function

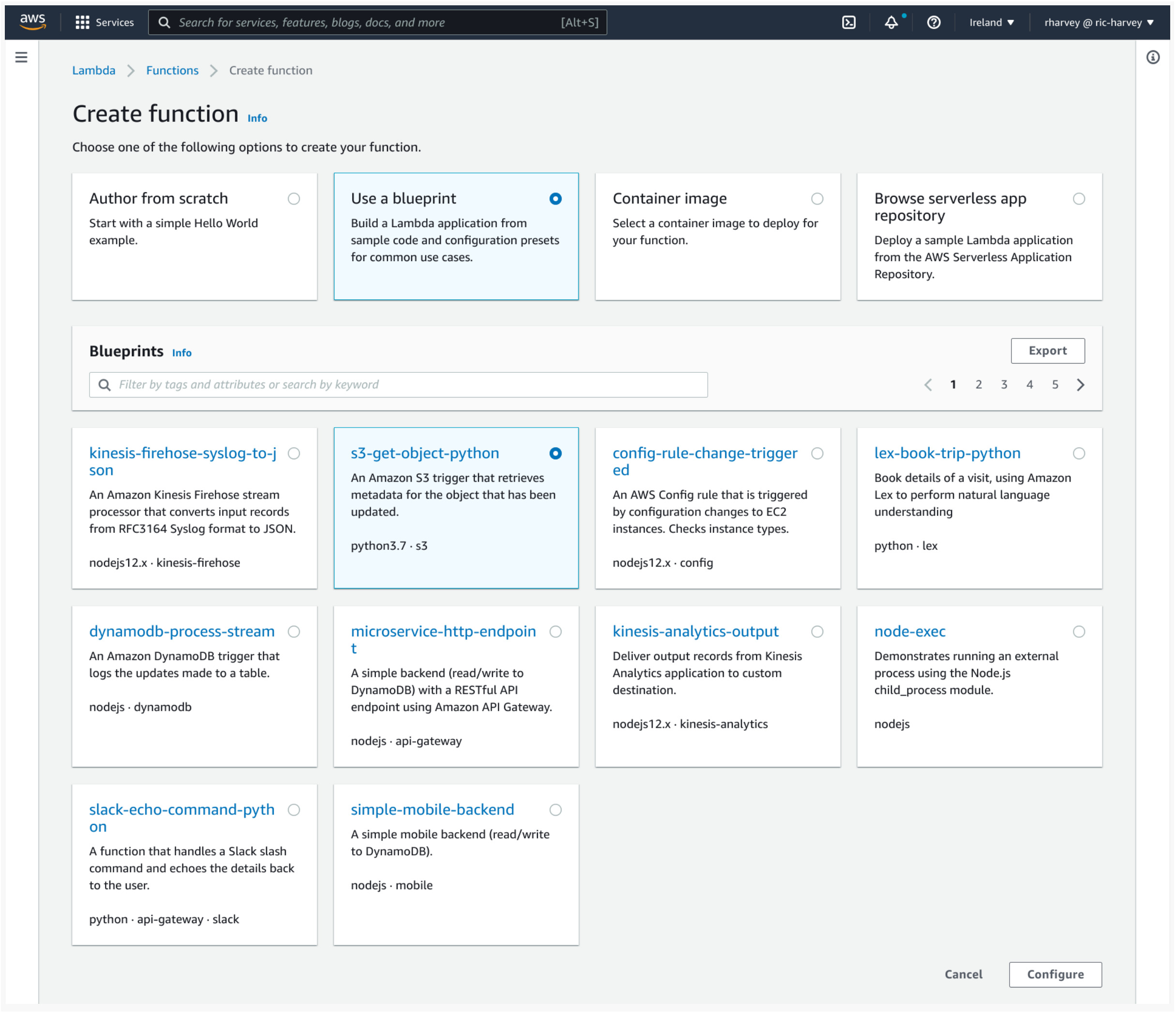
We are going to use a blueprint from AWS for the demo to get going quickly, but you can always adapt this to your needs. Blueprints not only contain the code but also include the scafolding for setting up the trigger from the S3 bucket to the function.

We are going to use a Python Blueprint from the create new function screen in the lambda console.


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
rharvey @ ric-harvey
Lambda > Functions
Functions (1)
Last fetched 20 seconds ago
Actions
Create function
Filter by tags and attributes or search by keyword

Function name	Description	Package type	Runtime	Code size	Last modified
s3redirect	Create redirects in S3	Zip	Python 3.8	1.3 kB	2 months ago

Now select *use a blueprint* and select the **s3-get-object-python** function. This is a python 3.7 example but the latest runtime of python you can create yourself currently goes up to 3.9. Now click configue:



aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
rharvey @ ric-harvey
Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint [✓]
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy for your function.

Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Blueprints Info

Filter by tags and attributes or search by keyword

kinesis-firehose-syslog-to-json
An Amazon Kinesis Firehose stream processor that converts input records from RFC3164 Syslog format to JSON.
nodejs12.x · kinesis-firehose

s3-get-object-python [✓]
An Amazon S3 trigger that retrieves metadata for the object that has been updated.
python3.7 · s3

config-rule-change-triggered
An AWS Config rule that is triggered by configuration changes to EC2 instances. Checks instance types.
nodejs12.x · config

lex-book-trip-python
Book details of a visit, using Amazon Lex to perform natural language understanding
python · lex

dynamodb-process-stream
An Amazon DynamoDB trigger that logs the updates made to a table.
nodejs · dynamodb

microservice-http-endpoint
A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway.
nodejs · api-gateway

kinesis-analytics-output
Deliver output records from Kinesis Analytics application to custom destination.
nodejs12.x · kinesis-analytics

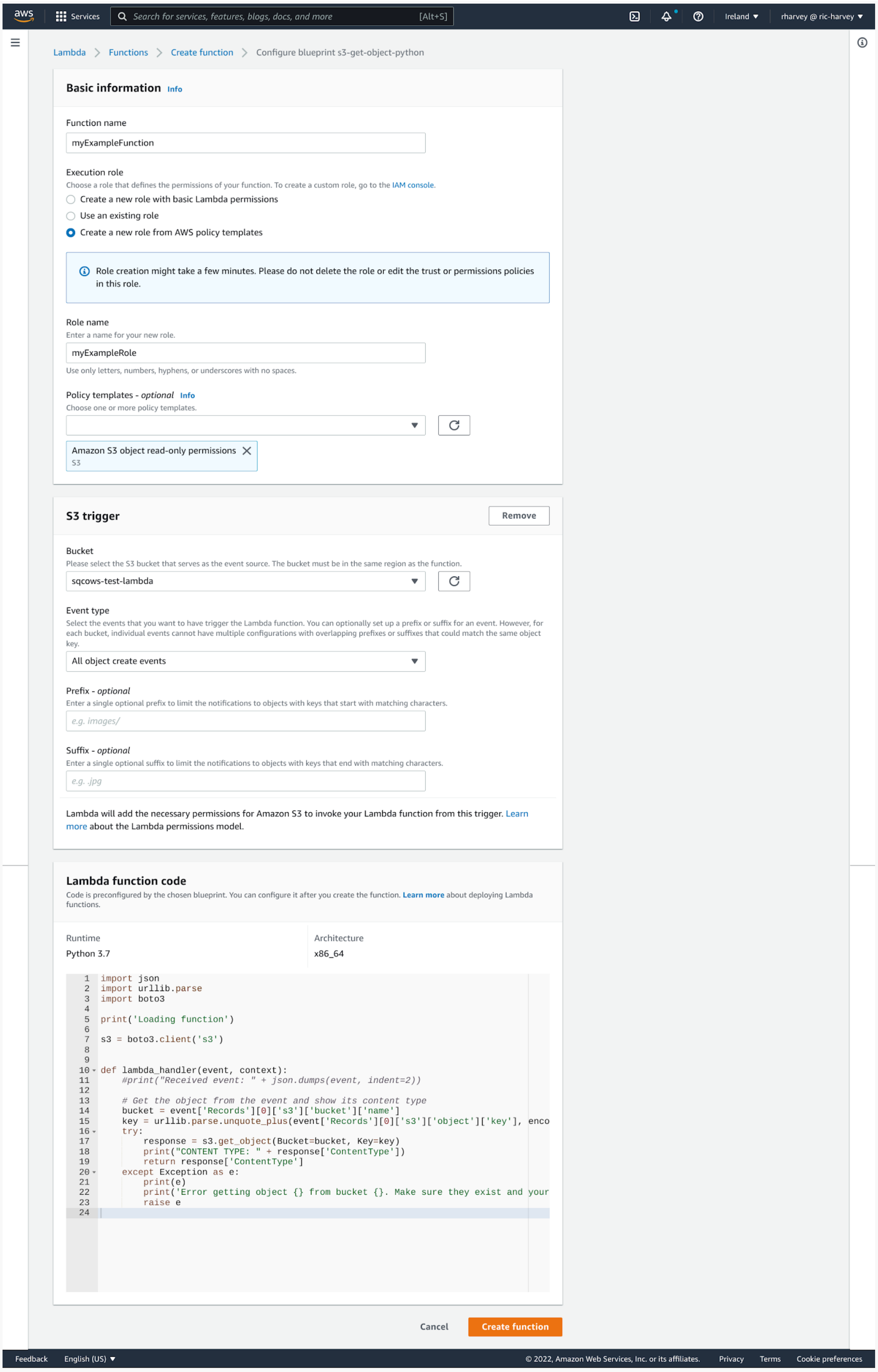
node-exec
Demonstrates running an external process using the Node.js child_process module.
nodejs

slack-echo-command-python
A function that handles a Slack slash command and echoes the details back to the user.
python · api-gateway · slack

simple-mobile-backend
A simple mobile backend (read/write to DynamoDB).
nodejs · mobile

- Under Basic information, do the following:
- For Function name, enter **myExampleFunction**
- For Execution role, choose Create a new role from AWS policy templates.
- For Role name, enter **myExampleRole**
- Under S3 trigger, choose the that you created with the aws cli.
    - The AWS console will automatically allow the function to access the S3 resource and most importantly allows S3 to tigger the function.
- Choose Create function.

Your screen should look like the following:


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Ireland
rharvey @ ric-harvey

Lambda > Functions > Create function > Configure blueprint s3-get-object-python

Basic information Info

Function name

myExampleFunction

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- ○ Create a new role with basic Lambda permissions
- ○ Use an existing role
- ● Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name

Enter a name for your new role.

myExampleRole

Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - *optional* Info

Choose one or more policy templates.

Amazon S3 object read-only permissions
S3

S3 trigger

Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

sqcows-test-lambda

Event type

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events

Prefix - *optional*

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

e.g. images/

Suffix - *optional*

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

e.g. .jpg

Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Lambda function code

Code is preconfigured by the chosen blueprint. You can configure it after you create the function. **Learn more** about deploying Lambda functions.

Runtime	Architecture
Python 3.7	x86_64

```
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], enco
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your
        raise e

```

Review the code

When the Lambda is triggered it recieves an *event*. Our function inspects the event and extracts the bucket name and the key (the key is the file name). Using the bucket name and key the function then uses boto3 to get the object and then print out the object type in the log. Your code should look like the following in your AWS console:

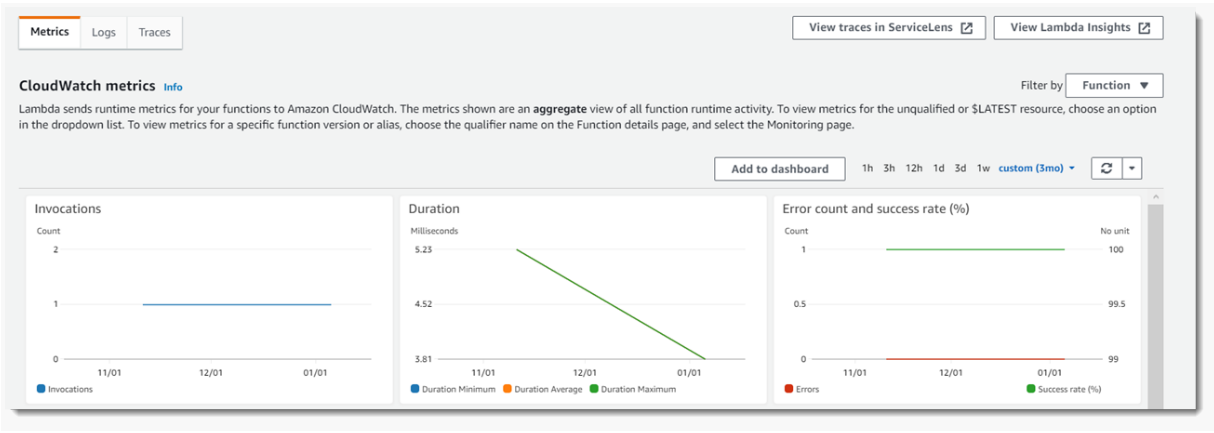
```
import json

import urllib.parse

import boto3



print('Loading function')



s3 = boto3.client('s3')





def lambda_handler(event, context):

    #print("Received event: " + json.dumps(event, indent=2))



    # Get the object from the event and show its content type

    bucket = event['Records'][0]['s3']['bucket']['name']

    key =
urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')

    try:

        response = s3.get_object(Bucket=bucket, Key=key)

        print("CONTENT TYPE: " + response['ContentType'])

        return response['ContentType']

    except Exception as e:

        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Test with the S3 trigger

Invoke your function when you upload a file to the Amazon S3 source bucket.

To test the Lambda function using the S3 trigger

- On the Buckets page of the Amazon S3 console, choose the name of the source that you created earlier.
- On the Upload page, upload a few .jpg or .png image files to the bucket.
- Open the Functions page of the Lambda console.
- Choose the name of your function (**myExampleFunction**).
- To verify that the function ran once for each file that you uploaded, choose the Monitor tab. This page shows graphs for the metrics that Lambda sends to CloudWatch. The count in the Invocations graph should match the number of files that you uploaded to the Amazon S3 bucket.



CloudWatch metrics Info

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an **aggregate** view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose an option in the dropdown list. To view metrics for a specific function version or alias, choose the qualifier name on the Function details page, and select the Monitoring page.

- You can also check out the logs in the CloudWatch Console or in the Lambda Console additionally and check out the log stream for you function name.

Clean up your resources

To clean up all the resources you can follow the steps below. This will ensure your account is nice and tidy and avoid a really really tiny storage cost fee of your uploaded trigger. Because the function is serverless, unless you upload more files to that S3 bucket you won’t be charged anything for it! This also means you are not wasting energy running a server waiting for something to happen.

To delete the Lambda function

- Open the Functions page of the Lambda console.
- Select the function that you created.
- Choose Actions, then choose Delete.
- Choose Delete.

To delete the IAM policy

- Open the Policies page of the AWS Identity and Access Management (IAM) console.
- Select the policy that Lambda created for you. The policy name begins with AWSLambdaS3ExecutionRole-.
- Choose Policy actions, Delete.
- Choose Delete.

To delete the execution role

- Open the Roles page of the IAM console.
- Select the execution role that you created.
- Choose Delete role.
- Choose Yes, delete.

To delete the S3 bucket

- Open the Amazon S3 console.
- Select the bucket you created.
- Choose Delete.
- Enter the name of the bucket in the text box.
- Choose Confirm.

Choose ARM

As a side note the function we created above used a x86 (intel) processor and is the default for all functions on AWS. However AWS have developed their own silicon chip called the Graviton Processsor and in Lambda you can currently use the v2 of this chip. It’s an ARM based processor and like it’s mobile phone cousin chips it sips electricity. Because of this you’ll use far less energy and it’s cheaper too!