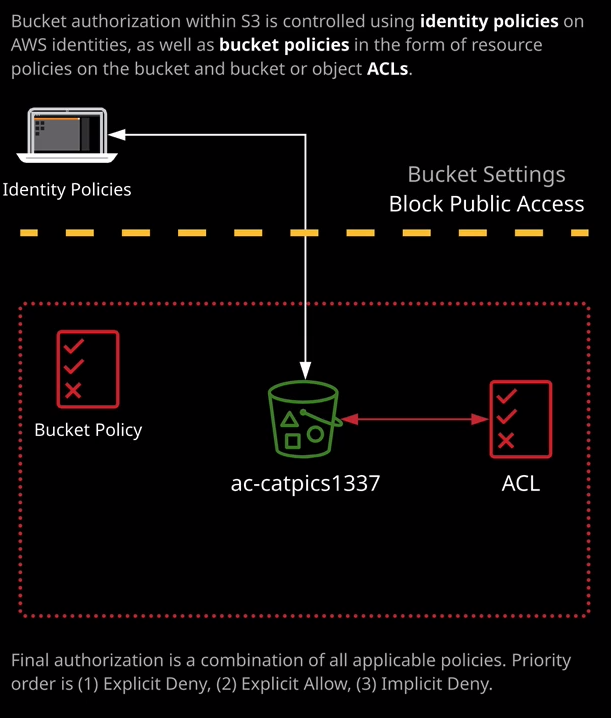
**S3 permissions**

S3 has a pretty complex permission structure, partly because it's a public facing service that requires a strong security foundation and it's one of the oldest AWS services. So it has a lot of legacy security baggage. Now the starting point for every **S3 bucket is that it's owned by the account that it's created in. That account has full control of that bucket. The bucket trusts the account that it's created in** now it's a critical concept to understand for the exam. **The foundation of S3 security is that the only entity that initially has access to a bucket is the account that creates the bucket.** Now that's the starting point for permissions. **The bucket by default doesn't trust any other AWS accounts, and it doesn't allow public access. You'd only trust the account that is created in.**

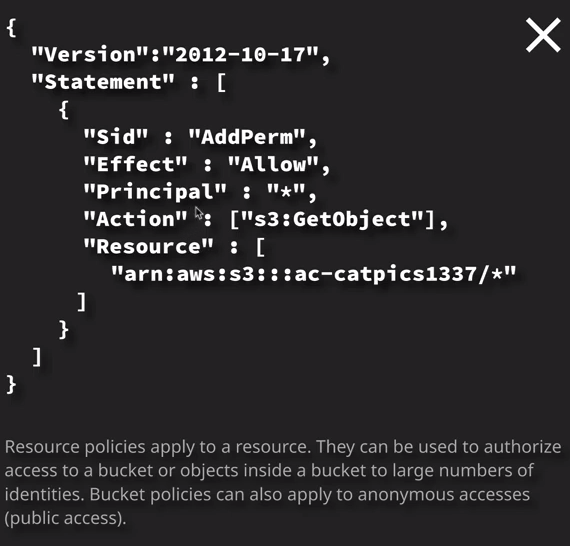


Now earlier in the course you learned **about identity policies** so these are policies are attached to AWS identities. This is a pretty common example. This identity policy contains three individual statements. The first allows the IAM user change the password. The second statement allows the IAM user to list all of the buckets so list all my buckets action on star resource and then this third statement allows the identity to do an S3:list star. So that's a wildcard as well as an S3:get\*. So both of these are wildcards and they allow this identity to do list\* operations and get\* operations both on this buckets so ac-catpics1337 as well as any objects inside that bucket. So this is a common identity policy that you'll find attached to an identity in AWS that gives base several permissions to change the password as well as basic level interactions with the bucket. So let's look at how this works in practice. Now I'm currently logged in to the root user for this AWS account. This is not something that I generally do, but I want to specifically illustrate how the root user interacts from a security perspective. So I'm going to move to the S3 console and I'm going to start by creating this S3 bucket that I've used in this example. So I'm going to create ac-catpics1337. So I go to create bucket and I'll give it the bucket name on the diagram to the right. Hopefully nobody else has created a bucket with this name. So I'll click on next I'll accept the rest of the defaults and create the bucket. Now because I'm logged in as the account root user and this bucket trust this AWS account. I'm able to access the bucket and I can interact with it. For example, I could go ahead and create a folder, so I'll create a folder in this bucket called Test and I'll hit save. So that demonstrates that at least I can interact with this bucket in a simple way. Next I want to do is log in using an IAM user that doesn't have any permissions on this account. Now, I've already gone ahead and created one. So if I click on the dropdown and go to IAM I have created a user in this account called S3 test user. So I'll go to users and there we go S3 test user. So I want to log into this account using that user. So I'm going to copy the IAM user's sign on link. I'm going to paste this into the address bar and open up the IAM logging console for this account. Once I'm there I'll log in as S3 test user and click on sign in. So we know that this user we're using because on the dropdown at the top, it says S3 test user and then the account alias for this account. Now, to demonstrate this if I go to the S3 console, I won't see the bucket because I don't have the permissions to list buckets and I won't be able to interact with the bucket directly, even if I could see it. **Now the reason for this is that IAM uses by default, have no access rights over this account, so they have no associated identity policies and because the only thing that this bucket trusts is the account and because the account doesn't delegate any permissions to this IAM user because it doesn't have any identity policies. It has no access over this bucket.**

So let's rectify that. So I'm going to click on the dropdown and I'm going to sign out, and I'm going to log in with my admin IAM user. So I'll go to the AWS management console link it should return me back to the IAM link for this account and it does. So I'm going to log in using an admin user. So if I go to the IAM console and I attach an identity policy to this user. So I'll locate the user. I'll make it an inline policy for now, just to make easier to understand. So I'll click on add inline policy. I'll click on choose service, select S3. Just to make things simple, I'll allocate the list and read sets of permissions to this user and then for resources I'm going to grant access on all resources in this account. So this IAM user has been given access to list and get actions for any resources in this account. So I'm going to review the policy. I'm fine with it. So I'll say S3 list and get. I'll create the policy. Make sure it's attached to the account. Everything looks good. I'll go back to the dashboard, and this time I'll log back in with this S3 test user and just verify that the permissions have changed. So I'm going to paste this URL into my address bar and then log back in with S3 test user this time, if I go to the S3 console I'll be able to see all of the buckets, including ac-catpicks1337. I'll be able to go into that bucket, interact with any objects or folders in that bucket, but if I attempt to create a folder, so I'll call this one test two and click on Save, I'll get permissions denied and that's because I don't have the relevant permissions the required permissions to create items so folders or objects inside this bucket. So for the exam, **it's important to understand that you'd use identity policies to grant access to S3 resources in the same account. So identity policies could be applied to roles, groups, and users and all of those three entities can be given permissions over S3 resources by creating an identity policy and then attaching that policy either as an inline policy or a managed policy to the identity. So roles, groups, and users, but critically, for the exam, you can only give permissions on S3 using identity policies to identities in your account. Identity policies can't be used to give other AWS accounts access to an S3 bucket or to give anonymous identities access to an S3 bucket. You can't assign an identity policy to an identity that you don't control. So because there's no way to reference anonymous identities so public people accessing your bucket logically, you can't apply an identity policy to the identity.**



So what if you want to apply various different access permissions to identities that you don't control? Well, there's a number of ways that you can do that. The first way is to use a **resource policy**. Now, resource policy is essentially just like an identity policy, but instead of being associated with an identity, it's associated with a resource such as S3. When resource policies that you use specifically with S3 it's known as a **bucket policy**. **Now bucket policies are applied to any identities accessing that bucket. These could be identities in your account, identities in other accounts, and even unauthenticated or anonymous users.** Now, this is what a bucket policy looks like. So again, **bucket policies are essentially resource policies that are applied to an S3 bucket and these policies apply to any identities accessing that bucket. So if you define something in a resource policy, regardless of where the identity comes from your account, other accounts, or anonymous, the permissions are applied or the applicable permissions are applied by the resource policy. They take effect.**

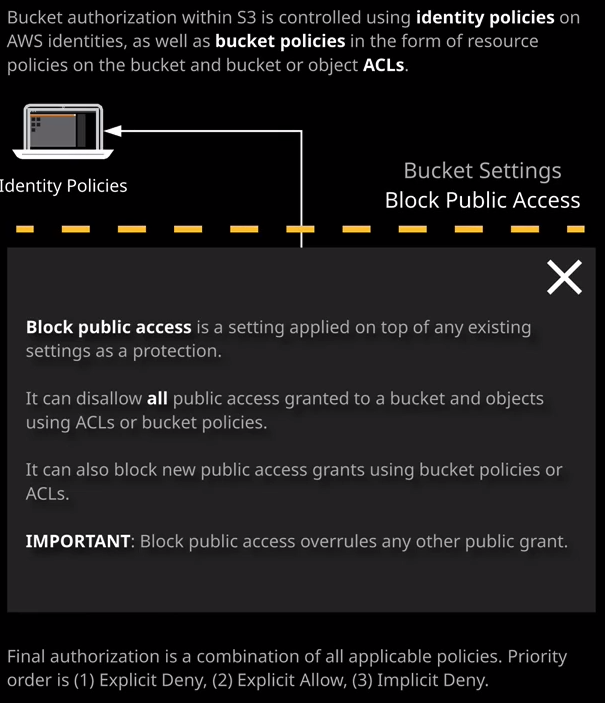


So in this example, this is a simple resource policy that could be applied to the ac-catpicks1337 S3 bucket that essentially allows any principle so any identity to be able to perform S3:getobject actions on any resources in this bucket and essentially this is the resource policy that is used to grant read only access to all objects in this bucket to anonymous identities. **Now bucket policies are applied logically at a bucket level.**

So if we select a bucket and we go to permissions and then go to the bucket policy selection, it's here where you can specify a bucket policy. So we could enter this bucket policy into this editor and apply public read only permissions to this bucket. Now that's something that you would only do when you intentionally want a bucket to be publicly accessible and you need to be especially wary when you're doing this because it's one very common way where you can actually cause data to be leaked unintentionally to the outside world. **So you need to be really careful whenever you're applying a bucket policy that grants public access. Now, remember, all policies inside AWS follow the priority rule. So if there are any policies that apply that explicitly deny something that always takes priority. If there isn't an explicit deny, then as long as you're explicitly allowed, then you can access whatever is being referenced. If there isn't an explicit allow or an explicit deny then you're implicitly denied, and if you're accessing a bucket using an identity and that bucket also has a resource policy then both of those policies are considered, so any explicit denies on either of those will take effect.**

So so far **we've got identity policies, which apply to identities in AWS. We've got bucket policies, which only apply to buckets, but you're able to grant access to specific objects using a bucket policy but there's only ever one bucket policy that's applied to a specific bucket.**

Now finally, we've got what's known as an **ACL or access control lists**. Now these are legacy methods of granting access to S3 buckets and objects. AWS no longer recommend using ACLs. They suggest using bucket or identity policies for anything new. **ACLs are what's known as a subresource. They're attached to objects on buckets and they control access in a very simple way to that object or bucket.** So to set ACLs at a bucket level, you can go to permissions and then go to access control list on the first item that's always on every AWS bucket is this ACL that grants access to the AWS account that you create that bucket in and that's a permission that can't be changed. This is what allows the bucket to trust the account. So by default, your account—your AWS account is given access to a list objects, write objects, to read bucket permissions, and write bucket permission. So that's what grants the trust between the bucket and the AWS account. In addition, you're able to use ACLs to grant public access to a bucket and also allow various other AWS entities such as the S3 log delivery group that we'll talk about later in the course. So I only want to mention this Just so you're aware of ACLs. Generally, you won't be asked to understand ACLs end to end anymore, because AWS are recommending not to use them but I want you to be aware that they do exist and ACLs can also be set on an object level. So if I go back to the bucket so ac-catpics1337 and let's say I upload an object to this bucket, so I'll go ahead and do that now. So I've selected object called oops.jpg. I'm going to accept the rest of the defaults and go ahead and upload that objects that will take a couple of seconds. Once it's completed, it will be visible inside the bucket. So if I select the object and then click on permissions. It's here, where I can set the ACLs on this particular object. **So ACLs are the only way that you can directly allocate security permissions to an object inside an S3 bucket. So if you want to be really granular and real control access to a specific object at the object level, then you'd need to utilize ACLs.** So if I wanted to grant public read only permissions to this specific object then I could do so by selecting everyone selecting read object and then hitting save and historically that would have defined the read only permission for public access at the object level but this one final thing that I want to draw your attention to, and that's a relatively new set of security settings available in S3.



Now when you create an S3 bucket and I'll just step through this again just to illustrate how this works. So let's say I make a new bucket and I call it ac-catpics1338. So I'll click on next. I'm not going to create this, but I just want to illustrate some of the settings. Click next again and you'll note that now, any creating an S3 bucket, you'll see this **block public access area**. Now by default, when you're creating a bucket S3 adds another layer of protections over the top, which is block public access. Now, this means by default, you won't be able to allow public access to the bucket or any objects in that bucket while this option is set, you can turn it off completely, which will mean you'll be able to grant public access as you require and you should only use this generally if you are an experience user of S3 or optionally, you can go ahead and allocate some public permissions now and then turn this option on to block any future settings. So if you've got a particular type of public access that you require then what you can do is you can set that public access and then tick these boxes that won't allow any new what public access grants, either using access control this or public bucket policy so that's something important to keep in mind. If you are granting public permissions, to any S3 buckets, you need to be aware that by default, public access is now blocked and in order to grant public access, you need to make sure that you configure that public access appropriately and then grant the public access.

**In summary, though, you'd use identity policies when you control the identity. When you want to grant access for a specific identity to one or more AWS objects and you're doing that from the focus point of an identity. You'd use a bucket policy when you want to grant access to a specific bucket to more than one identity, maybe in your own account, maybe other AWS accounts or public access. So the decision between an identity policy and a resource policy depends on the perspective that you're creating these permissions. Generally, though, you would not use ACLs outside a very specific set of scenarios and AWS themselves at this point, don't recommend using ACLs for any new deployments.**

<https://aws.amazon.com/blogs/security/iam-policies-and-bucket-policies-and-acls-oh-my-controlling-access-to-s3-resources/>

**When to use IAM policies vs. S3 policies**

Use IAM policies if:

* You need to control access to AWS services other than S3. IAM policies will be easier to manage since you can centrally manage all of your permissions in IAM, instead of spreading them between IAM and S3.
* You have numerous S3 buckets each with different permissions requirements. IAM policies will be easier to manage since you don’t have to define a large number of S3 bucket policies and can instead rely on fewer, more detailed IAM policies.
* You prefer to keep access control policies in the IAM environment.

Use S3 bucket policies if:

* You want a simple way to grant [cross-account access](http://docs.aws.amazon.com/AmazonS3/latest/dev/AccessPolicyLanguage_UseCases_s3_a.html) to your S3 environment, without using [IAM roles](http://docs.aws.amazon.com/IAM/latest/UserGuide/cross-acct-access-walkthrough.html).
* Your IAM policies bump up against the size limit (up to 2 kb for users, 5 kb for groups, and 10 kb for roles). S3 supports bucket policies of up 20 kb.
* You prefer to keep access control policies in the S3 environment.

If you’re still unsure of which to use, consider which audit question is most important to you:

* If you’re more interested in “What can this user do in AWS?” then IAM policies are probably the way to go. You can easily answer this by looking up an IAM user and then examining their IAM policies to see what rights they have.
* If you’re more interested in “Who can access this S3 bucket?” then S3 bucket policies will likely suit you better. You can easily answer this by looking up a bucket and examining the bucket policy.

Whichever method you choose, we recommend staying as consistent as possible. Auditing permissions becomes more challenging as the number of IAM policies and S3 bucket policies grows.