Beyond “Read All”:

BUILD FINE-GRAINED CONTROL OF AMAZON WEB SERVICES IN YOUR CFML APP

Brian Klaas
@brian_klaas

AWS
Simple Storage Service


STORE ALL THE THINGS!

```
cffile( variable="fileData", file="s3://somebucket/somefile.txt", action="read" );
```

```
cfdirectory( directory="s3://somebucket/someDirectory", action="list" );
```


IT'S ALMOST TOO EASY

“principal”: “*”

Verizon

Names, addresses, account details, and account personal identification numbers (PINs) of as many as 14 million US customers.

HTTPS://WWW.UPGUARD.COM/BREACHES/VERIZON-CLOUD-LEAK

Dow Jones

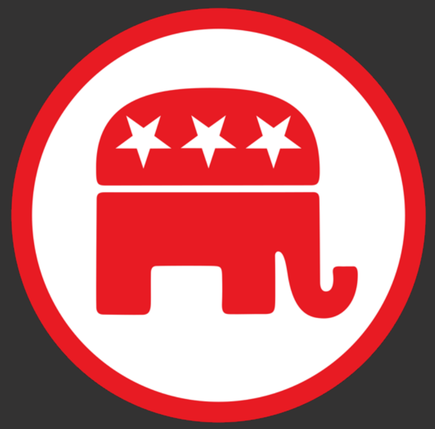
Sensitive personal and financial details of 2.2 million customers.

HTTPS://WWW.UPGUARD.COM/BREACHES/CLOUD-LEAK-DOW-JONES

FedEx

Customer passports, driver licenses.

HTTPS://ARSTECHNICA.COM/INFORMATION-TECHNOLOGY/2018/02/FEDEX-CUSTOMER-DATA-LEFT-ONLINE-FOR-ANYONE-TO-RIFLE-THROUGH/



Republican National Committee

200 million voter records.

HTTPS://WWW.SKYHIGHNETWORKS.COM/CLOUD-SECURITY-BLOG/VERIZON-DATA-BREACH-TWO-EASY-STEPS-TO-PREVENT-AWS-S3-LEAKS/

Macy's

Customer profiles, including address and date of birth.

HTTPS://WWW.DOJ.NH.GOV/CONSUMER/SECURITY-BREACHES/DOCUMENTS/MACYS-20180702.PDF

Booz | Allen | Hamilton

Booz Allen Hamilton

Files related to the National Geospatial-Intelligence Agency (NGA), which handles battlefield satellite and drone imagery.

National Credit Federation

11GB of credit card numbers, credit reports from the three major reporting agencies, bank account numbers and Social Security numbers.

HTTPS://BUSINESSINSIGHTS.BITDEFENDER.COM/WORST-AMAZON-BREACHES


BUILD ALL THE THINGS
NOT INFRASTRUCTURE!

Unsolicited, obvious advice:

Consider your authentication + authorization strategy before building.

AWS Identity Access Management

Beyond “Read All”:

BUILD FINE-GRAINED CONTROL OF AMAZON WEB SERVICES IN YOUR CFML APP

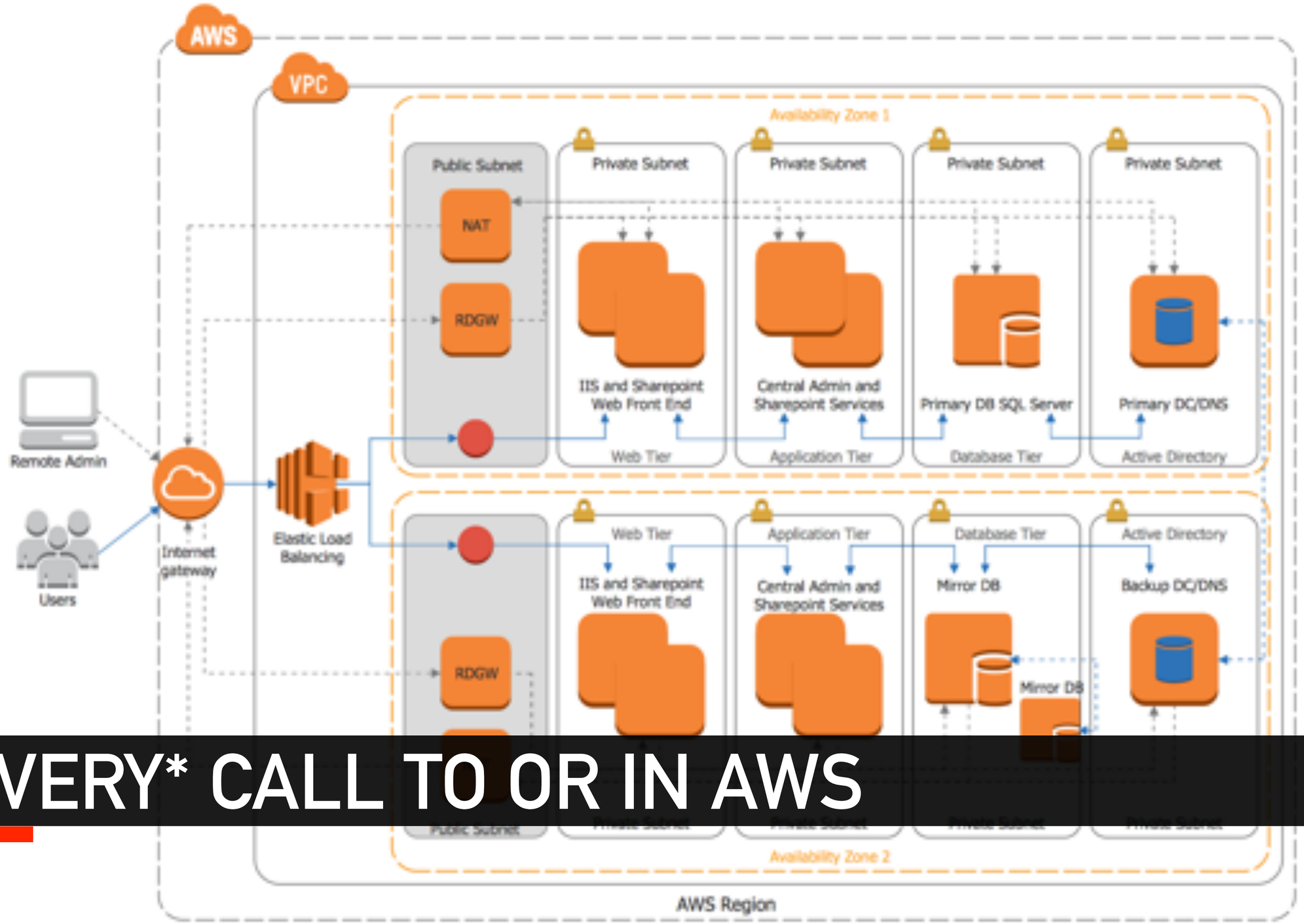
Brian Klaas
@brian_klaas


NOT COMPREHENSIVE

AWS Playbox App

https://github.com/brianklaas/awsPlaybox

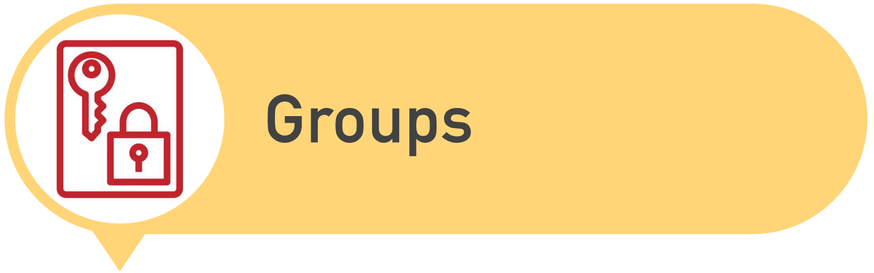
AWS Identity Access Management

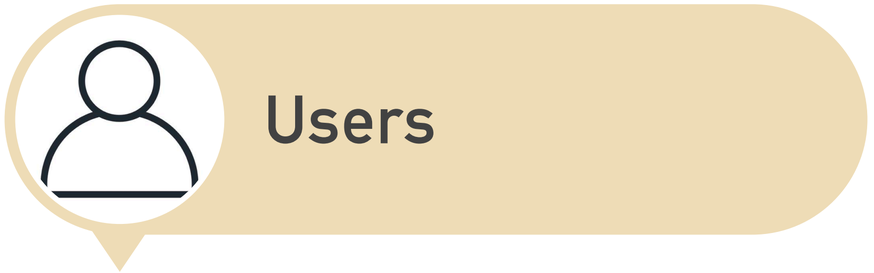


EVERY CALL TO OR IN AWS


Policies


Roles


Groups

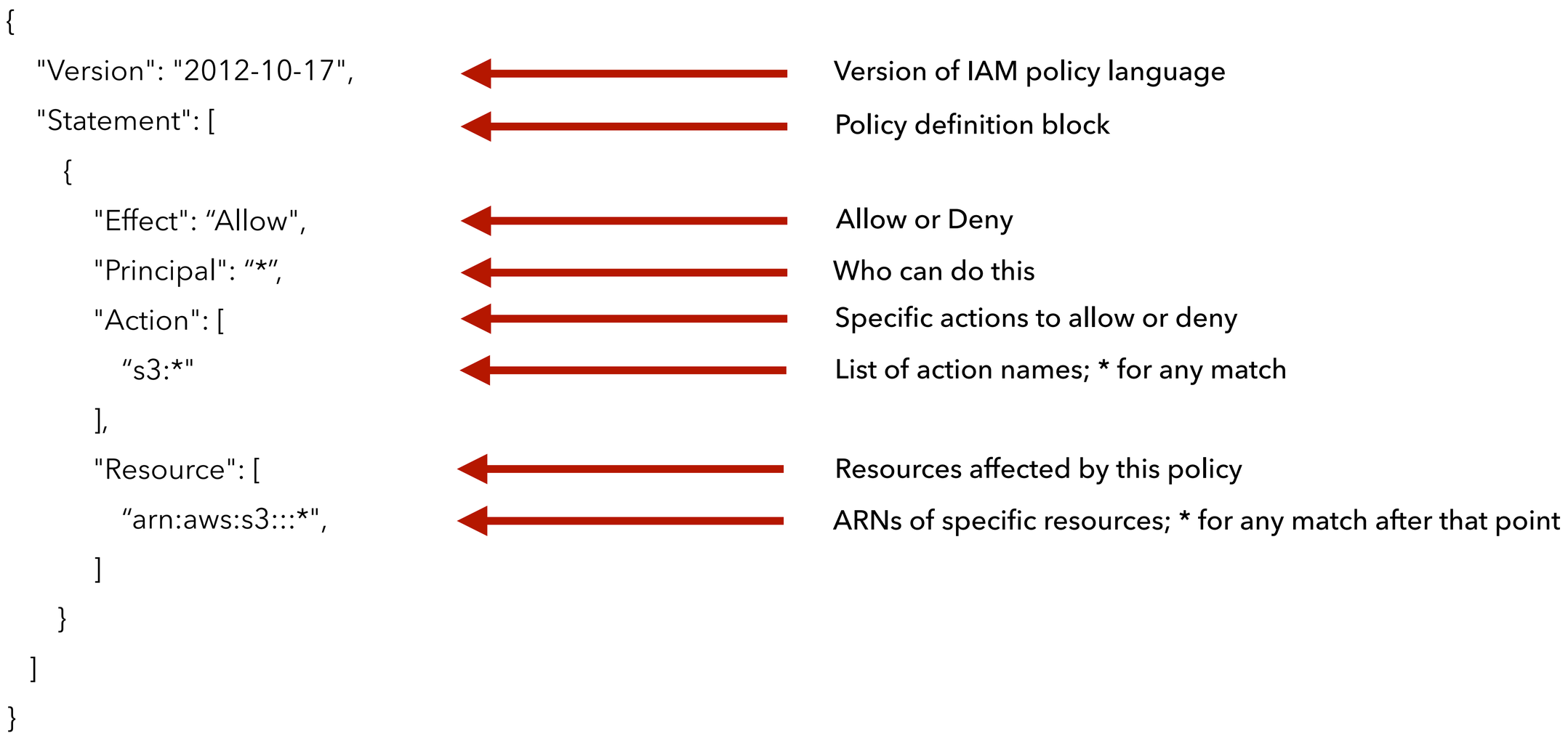

Users

Policies

Mastering IAM = mastering polices

Policies = JSON structures

Anatomy of a Policy



ARN = Amazon Resource Name

Example ARNs

S3 Bucket

arn:aws:s3:::awsplayboxbucket

CloudFormation Stack

arn:aws:cloudformation:us-east-1:0123456789:stack/awseb-e-kmjwp8btzp-stack/9e2c9e50-bcef-11e8-87f3-503aca261699

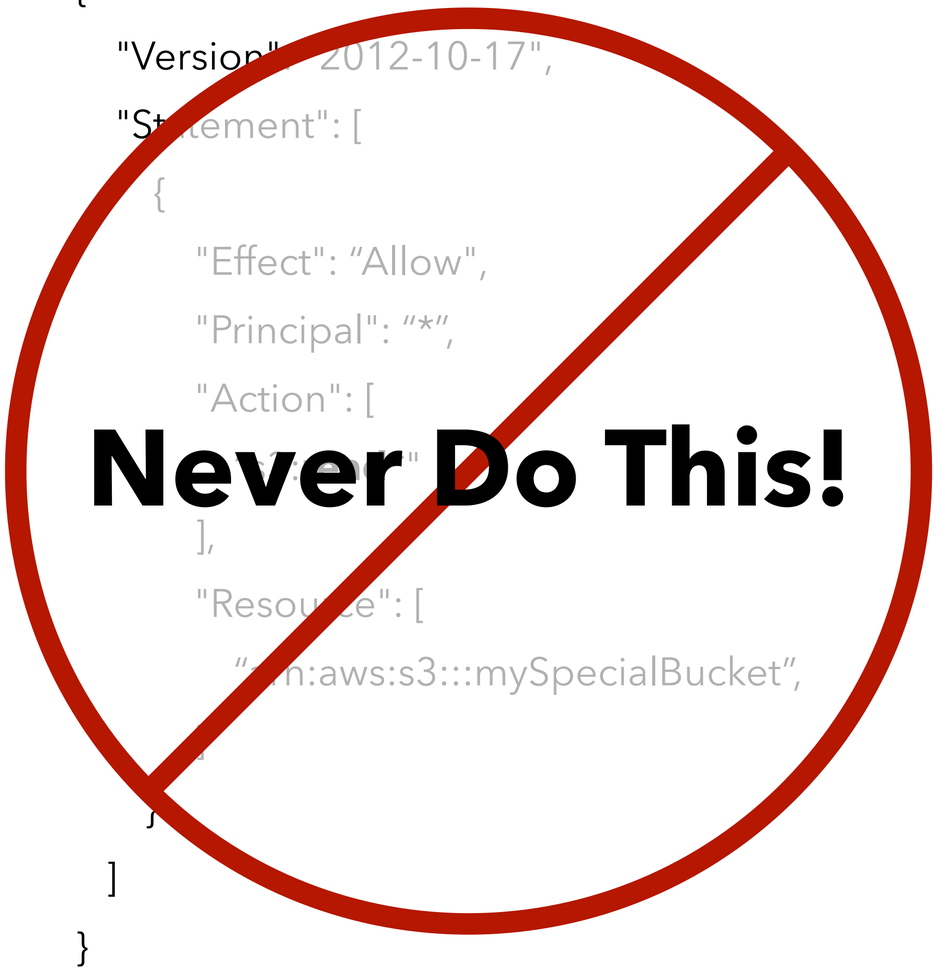
SNS Topic

arn:aws:sns:us-east-1:0123456789:AWSPlayboxDemoTopic-2019-02-20-14-48-38

Lambda Function

arn:aws:lambda:us-east-1:0123456789:function:confDemoSimpleJSONReturn

Anatomy of a Policy



Be specific.
Allow the least privilege that makes sense.

Basic Read/Write S3 Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
        "Effect": "Allow",
        "Action": [
           "s3:PutObject",
           "s3:GetObjectAcl",
           "s3:GetObject",
           "s3:ListBucket",
           "s3:GetBucketAcl",
           "s3:DeleteObject"
        ],
        "Resource": [
           "arn:aws:s3:::awsplayboxprivatebucket",
           "arn:aws:s3:::awsplayboxprivatebucket/*"
        ]
     } ] }
```

No principal =
Can apply to
multiple entities

Must specify the bucket and the items in the bucket

Restrict How a Service Is Called

```
{
   "Version": "2012-10-17",
   "Statement": [ {
      “Sid”: “Allow IAM user to publish to the SNS topic only if the request comes from a specific Lambda function.”,
      "Effect": "Allow",
      “Principal": { "AWS": “arn:aws:iam:0123456789:user/billingApp” },
      “Action": "sns:publish",
      “Resource": “arn:aws:sns:us-east-1:0123456789:billsPastDueTopic“,
      “Condition”: { “ArnEquals”: {“aws:SourceArn”: “arn:aws:lambda:us-east-1:0123456789:function:
checkForBillPastDue”} }
   } ]
}
```

Allow Write to DynamoDB During Tax Season

```
{
   "Version": "2012-10-17",
   "Statement": [ {
         "Effect": "Allow",
         “Action": [
              “dynamodb:GetItem”,
              “dynamodb:PutItem”,
              “dynamodb:UpdateItem"
         ],
         “Resource": “arn:aws:dynamodb:us-east-1:0123456789:table/customerIncome“,
         “Condition”: {
              “DateGreaterThan”: {“aws:CurrentTime”: “2019-04-01T04:00:00Z” },
              “DateLessThan”: {“aws:CurrentTime”: “2019-04-16T04:00:00Z” },
              “IpAddress”: {“aws:SourceIp”: [ “192.0.2.0/24”, “203.0.113.86” ] }
         }
      } ]
}
```

Ensure All S3 Requests Are Over https

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         “Action”: ”s3:*”,
         “Principal”: ”*”,
         “Resource”: “arn:aws:s3:::bucketname/*”,
         “Condition”: {
            “Bool”: { “aws:SecureTransport”: false }
         }
      } ]
}
```

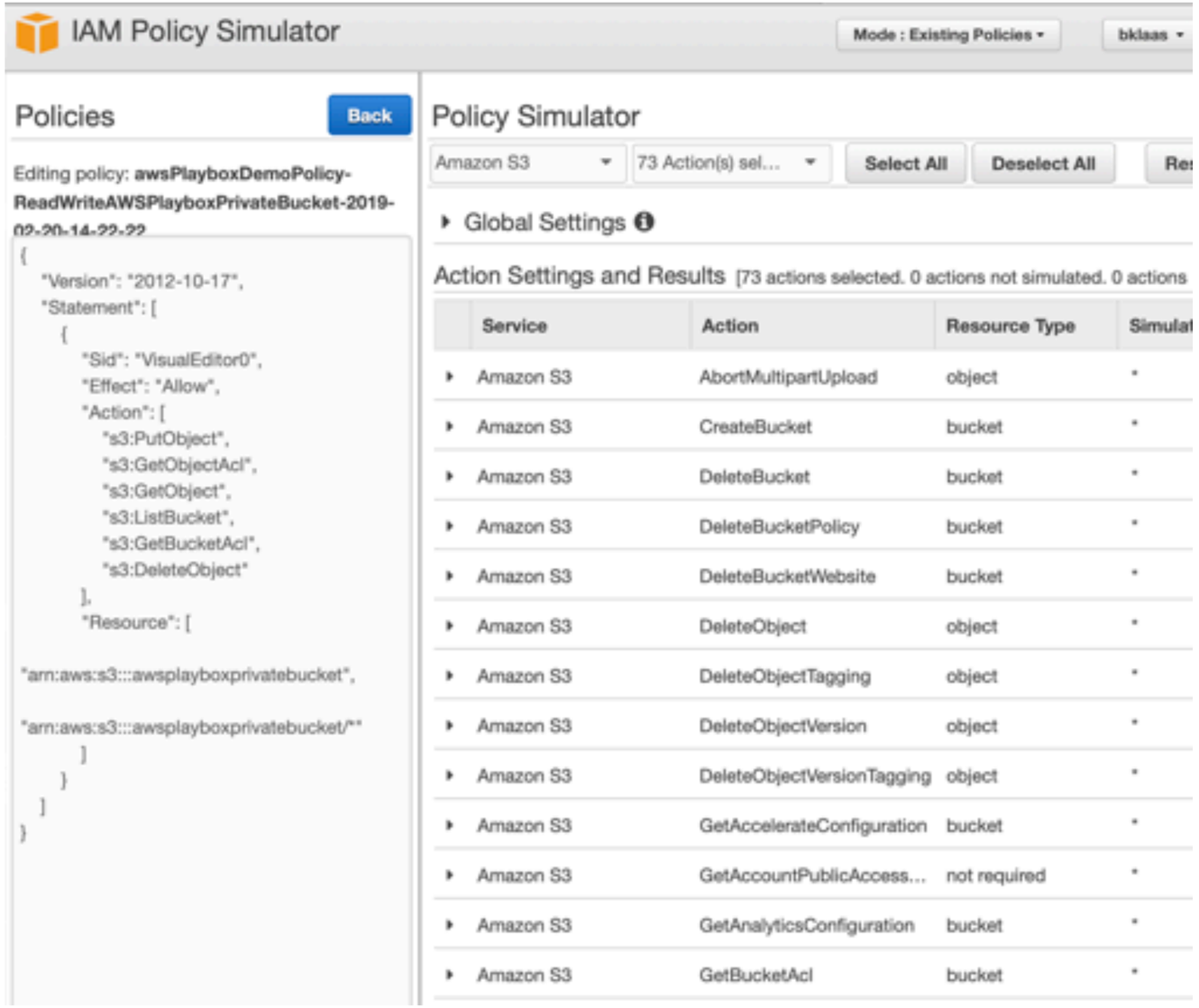
Read/Write All S3 Buckets Except One

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:GetObjectAcl",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:GetBucketAcl",
            "s3:DeleteObject"
         ],
         “NotResource": “arn:aws:s3:::security_audit_bucket/*”
      } ]
}
```

Mastering IAM = mastering polices

The IAM Policy Simulator

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html



Hello?

CFML?

Using the AWS Java SDK

Add to cfusion/lib:

- CF2018:
  - aws-java-sdk-1.11.xxx.jar
- Other runtimes: the SDK .jar, plus:
  - jackson-annotations-2.6.0.jar
  - jackson-core-2.6.7.jar
  - jackson-databind-2.6.7.1.jar
  - joda-time-2.8.1.jar

AWS Playbox App

https://github.com/brianklaas/awsPlaybox

Basic Pattern to Accessing the AWS Java SDK

1. Create a service object
2. Create a request object
3. Populate the attributes of the request object
4. Tell the service object to run a function on the request object
5. Get a result object back

Creating an IAM Policy

1. Create the **IAM** service object
2. Create a **createPolicyRequest** object
3. Populate the attributes of the **createPolicyRequest** object
4. Tell the IAM service object to **createPolicy(createPolicyRequest)**
5. Get a **createPolicyResult** object back

Let’s show some code!

```
iam = application.awsServiceFactory.createServiceObject(‘iam’);  1

policyName = 'awsPlayboxDemoPolicy-ReadWriteAWSPlayboxPrivateBucket';

createPolicyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreatePolicyRequest')  2

        .withPolicyName(policyName)  3

        .withDescription('Allows read/write permission to the awsPlayboxPrivate S3 bucket.');

policyJSON = fileRead(expandPath("./iamPolicies/awsPlayboxPrivateReadWrite.txt"));

createPolicyRequest.setPolicyDocument(policyJSON);

createPolicyResult = iam.createPolicy(createPolicyRequest);

policyDetails = createPolicyResult.getPolicy();

application.awsResources.iam.S3PolicyARN = policyDetails.getARN();
```

CODE: CREATING A POLICY

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificBucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObjectAcl",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:GetBucketAcl",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::awsplayboxprivatebucket",
        "arn:aws:s3:::awsplayboxprivatebucket/*"
      ]
    } ] }
```

```
iam = application.awsServiceFactory.createServiceObject('iam');
policyName = 'awsPlayboxDemoPolicy-ReadWriteAWSPlayboxPrivateBucket';
createPolicyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreatePolicyRequest')
        .withPolicyName(policyName)
        .withDescription('Allows read/write permission to the awsPlayboxPrivate S3 bucket.');
policyJSON = fileRead(expandPath("./iamPolicies/awsPlayboxPrivateReadWrite.txt"));
createPolicyRequest.setPolicyDocument(policyJSON);
createPolicyResult = iam.createPolicy(createPolicyRequest); 4
policyDetails = createPolicyResult.getPolicy(); 5
application.awsResources.iam.S3PolicyARN = policyDetails.getARN();
```

What if we don’t know the resource name or ARN in advance?

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowsPublishToOneTopic",
         "Effect": "Allow",
         "Action": "sns:Publish",
         "Resource": "%CURRENT_TOPIC_ARN%"
      }
   ]
}
```

```
iam = application.awsServiceFactory.createServiceObject(‘iam’);

policyName = 'awsPlayboxDemoPolicy-SendToSNS';

createPolicyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreatePolicyRequest')

        .withPolicyName(policyName)

        .withDescription('Allows user to send message to a specific SNS topic');

policyJSON = fileRead(expandPath("./iamPolicies/snsSendMessage.txt"));

policyJSON = replace(policyDetails, "%CURRENT_TOPIC_ARN%", application.awsResources.currentSNSTopicARN);

createPolicyRequest.setPolicyDocument(policyDetails);

createPolicyResult = iam.createPolicy(createPolicyRequest);

policyDetails = createPolicyResult.getPolicy();

application.awsResources.iam.SNSPolicyARN = policyDetails.getARN();
```

```
iam = application.awsServiceFactory.createServiceObject(‘iam’);
policyName = 'awsPlayboxDemoPolicy-SendToSNS';
createPolicyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreatePolicyRequest')
        .withPolicyName(policyName)
        .withDescription('Allows user to send message to a specific SNS topic');
policyJSON = fileRead(expandPath("./iamPolicies/snsSendMessage.txt"));
policyJSON = replace(policyDetails, "%CURRENT_TOPIC_ARN%", application.awsResources.currentSNSTopicARN);
createPolicyRequest.setPolicyDocument(policyDetails);
createPolicyResult = iam.createPolicy(createPolicyRequest);
policyDetails = createPolicyResult.getPolicy();
application.awsResources.iam.SNSPolicyARN = policyDetails.getARN();
```

```
iam = application.awsServiceFactory.createServiceObject(‘iam’);

policyName = 'awsPlayboxDemoPolicy-SendToSNS';

createPolicyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreatePolicyRequest')

        .withPolicyName(policyName)

        .withDescription('Allows user to send message to a specific SNS topic');

policyJSON = fileRead(expandPath("./iamPolicies/snsSendMessage.txt"));

policyJSON = replace(policyDetails, "%CURRENT_TOPIC_ARN%", application.awsResources.currentSNSTopicARN);

createPolicyRequest.setPolicyDocument(policyDetails);

createPolicyResult = iam.createPolicy(createPolicyRequest);

policyDetails = createPolicyResult.getPolicy();

application.awsResources.iam.SNSPolicyARN = policyDetails.getARN();
```

Learning More About Policies

AWS Docs

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html

Complete AWS IAM Reference

https://iam.cloudonaut.io

An Excellent Session from re:Invent

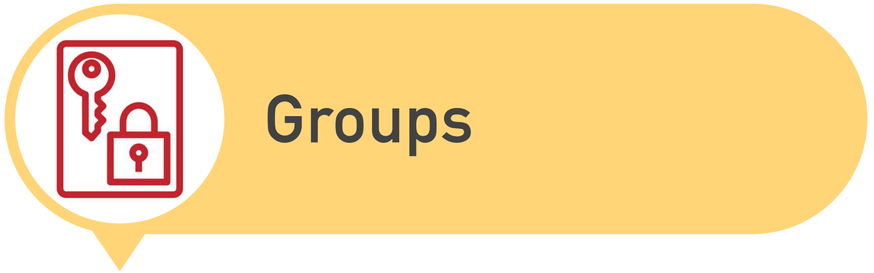
https://www.youtube.com/watch?v=YQsK4MtsELU

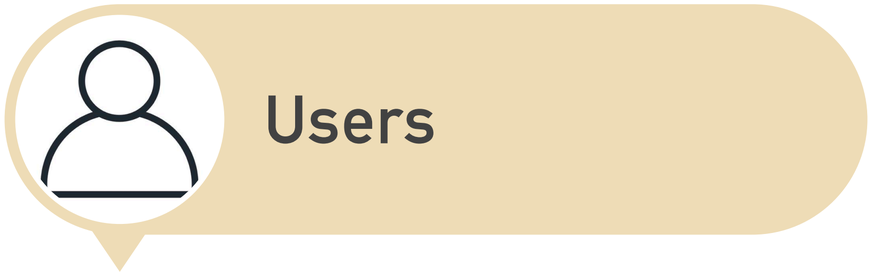
The Best Tutorial I've Found

https://start.jcolemorrison.com/aws-iam-policies-in-a-nutshell/


Policies


Roles


Groups

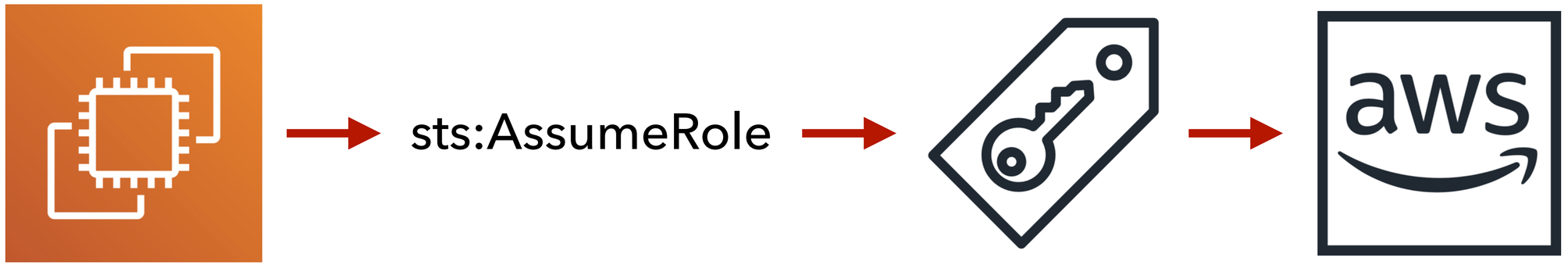

Users

Roles

Roles are not associated with a specific user or group.
Trusted entities assume roles.

Roles let you share access
without setting up access keys (users).

sts:AssumeRole


sts:AssumeRole
aws

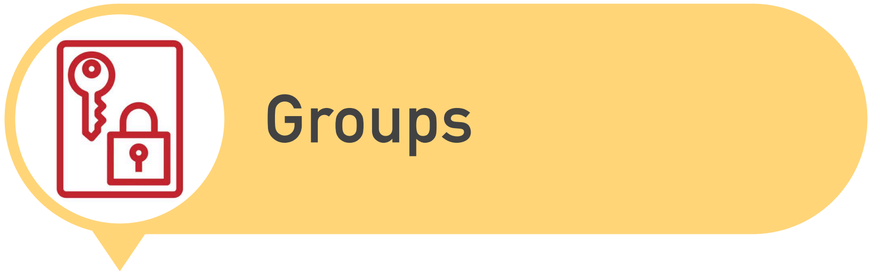
WARNING: Pseudocode!

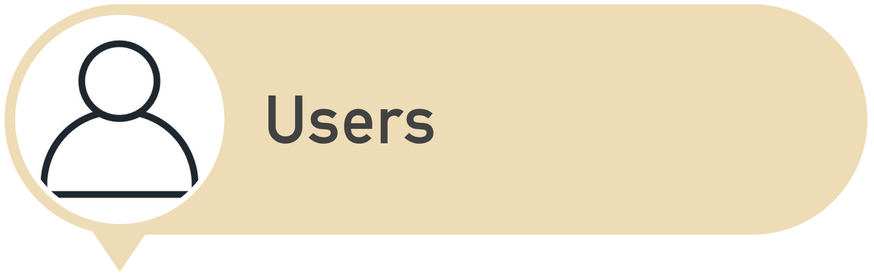
```
assumeRoleResult = AssumeRole(ARN of the role you need to assume);
tempCredentials = new SessionAWSCredentials(
    assumeRoleResult.AccessKeyId,
    assumeRoleResult.SecretAccessKey,
    assumeRoleResult.SessionToken);
s3Client = CreateAmazonS3Client(tempCredentials);
```

Roles = JSON structures


Policies


Roles


Groups


Users

Groups

Creating an IAM Group

1. Create the **IAM** service object
2. Create a **createGroupRequest** object
3. Populate the attributes of the **createGroupRequest** object
4. Tell the IAM service object to **createGroup(createGroupRequest)**
5. Get a **createGroupResult** object back

```
iam = application.awsServiceFactory.createServiceObject(‘iam');   1
groupName = 'awsPlayboxDemoGroup';
createGroupRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreateGroupRequest')   2
        .withGroupName(groupName);   3
createGroupResult = iam.createGroup(createGroupRequest);   4
groupDetails = createGroupResult.getGroup();   5
application.awsResources.iam.PlayboxGroupARN = groupDetails.getARN();
```

Attach policies to groups, not users!

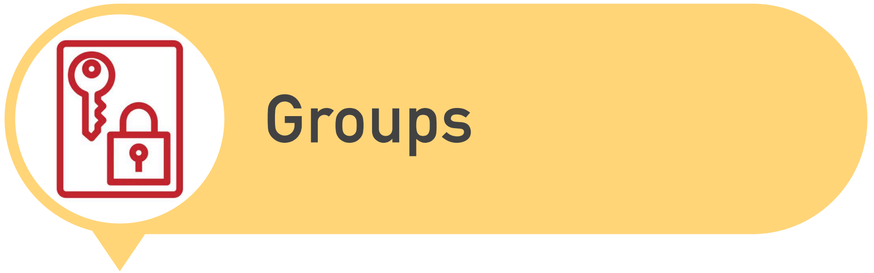
Attaching a Policy to a Group

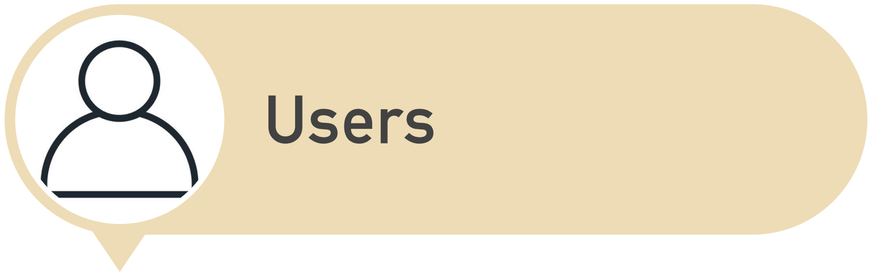
1. Create the **IAM** service object
2. Create a **attachGroupPolicyRequest** object
3. Populate the attributes of the **attachGroupPolicyRequest** object
4. Tell the IAM service object to **attachGroupPolicy(attachGroupPolicyRequest)**
5. Get a **attachGroupPolicyRequestResult** object back

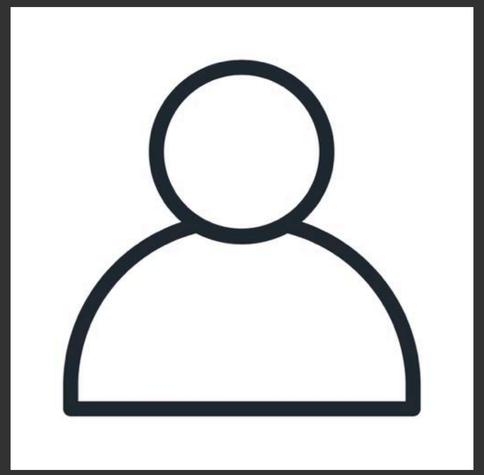
```
attachGroupPolicyRequest = CreateObject('java',
'com.amazonaws.services.identitymanagement.model.AttachGroupPolicyRequest')  2

        .withGroupName(groupName)  3

        .withPolicyArn(application.awsResources.iam.S3PolicyARN);

attachGroupPolicyRequestResult = iam.attachGroupPolicy(attachGroupPolicyRequest);  4  5



attachGroupPolicyRequest = CreateObject('java',
'com.amazonaws.services.identitymanagement.model.AttachGroupPolicyRequest')

        .withGroupName(groupName)

        .withPolicyArn(application.awsResources.iam.SNSPolicyARN);

attachGroupPolicyRequestResult = iam.attachGroupPolicy(attachGroupPolicyRequest);
```


Policies

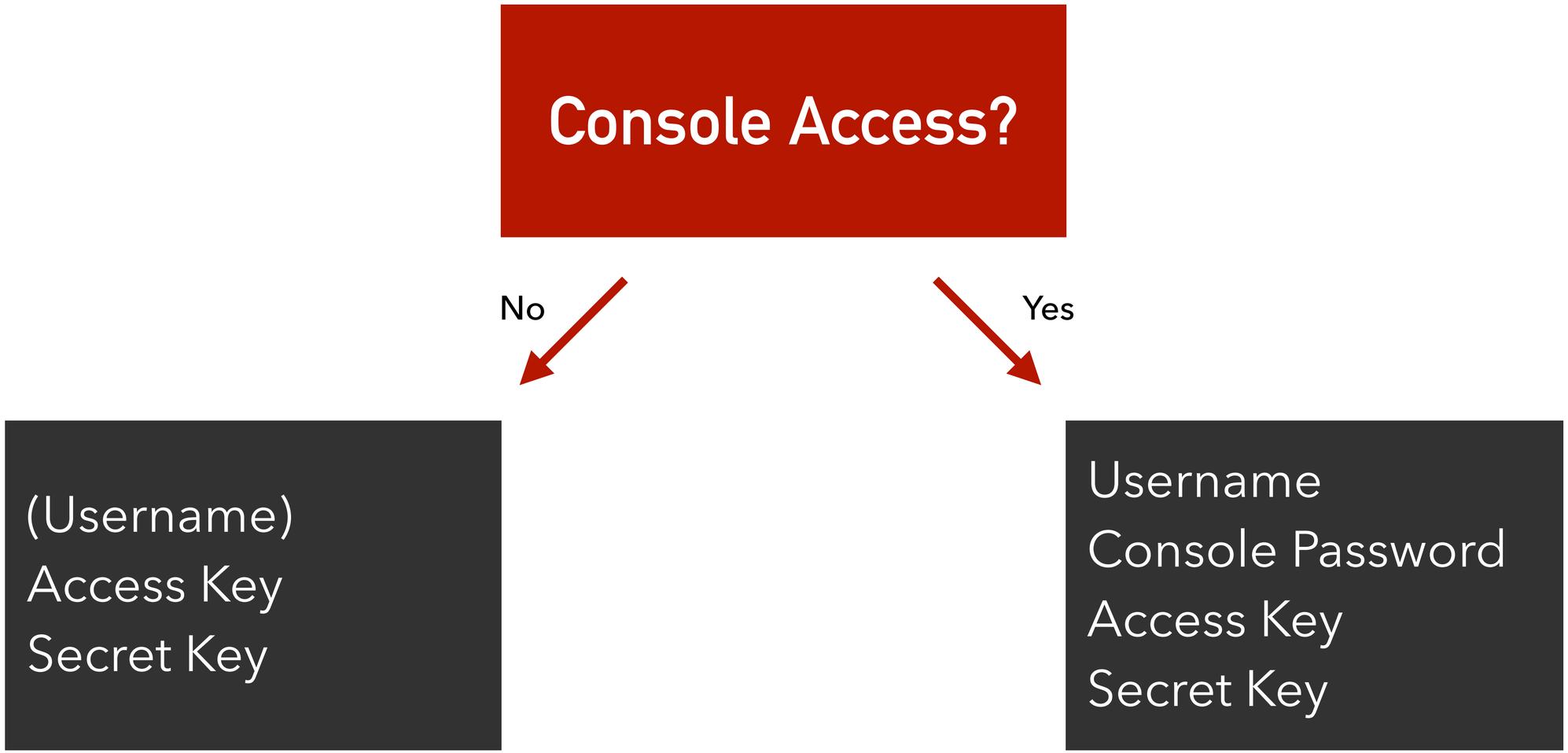

Roles


Groups


Users



Users


Console Access?
No
Yes
(Username)
Access Key
Secret Key
Username
Console Password
Access Key
Secret Key

Access Key
Secret Key

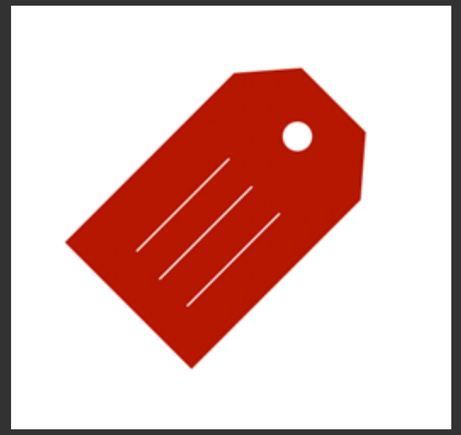
The User Creation Process

1. Create the user
2. Create the access key for the user
3. Add user to a group which has policies attached

Creating an IAM User

1. Create the **IAM** service object
2. Create a **createUserRequest** object
3. Populate the attributes of the **createUserRequest** object
4. Tell the IAM service object to **createUser(createUserRequest)**
5. Get a **createUserResult** object back

```
iam = application.awsServiceFactory.createServiceObject(‘iam');  1

userName = 'awsPlayboxDemoUser';

 createUserRequest = CreateObject('java',
'com.amazonaws.services.identitymanagement.model.CreateUserRequest')  2

            .withUserName(userName);  3

createUserResult = iam.createUser(createUserRequest);  4

userDetails = createUserResult.getUser();  5
```



Detour: Tags

Tags are for finding your stuff in AWS
in a human-readable way.

Types of Tags

- Key-value pairs
- User-defined
- Cost Allocation
- Can create Resource Groups based on tags

Business		Technical		Security	
Cost Center	41001	Environment	Dev	Compliance	HIPAA
Department	Security	Version	2.2.1	Data Sensitivity	4
Owner	Bill Bridges	Application	Cart	Encrypted	Yes

Tagging Best Practices

https://aws.amazon.com/answers/account-management/aws-tagging-strategies/

```
userTag = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.Tag')
            .withKey('department')
            .withValue(‘IT Security’);
tagArray = [ userTag ];
createUserRequest.setTags(tagArray);
```

The User Creation Process

1. Create the user 
2. Create the access key for the user
3. Add user to a group which has policies attached

Users have no credentials by default.
Users are not part of any group by default.

Creating User Credentials

1. Create the **IAM** service object
2. Create a **createAccessKeyRequest** object
3. Populate the username for the **createAccessKeyRequest** object
4. Tell the IAM service object to **createAccessKey(createAccessKeyRequest)**
5. Get a **createAccessKeyResult** object back

```
createAccessKeyRequest = CreateObject('java',
'com.amazonaws.services.identitymanagement.model.CreateAccessKeyRequest')  2

            .withUserName(userName);  3

createAccessKeyResult = iam.createAccessKey(createAccessKeyRequest);  4

accesKeyInfo = createAccessKeyResult.getAccessKey();  5

userAccessKey = accesKeyInfo.getAccessKeyID();

userSecretKey = accesKeyInfo.getSecretAccessKey();
```

There is no way
to retrieve a secret key
after it has been created.

You are fully responsible for the security of secret keys that you store locally.

The User Creation Process

1. Create the user 
2. Create the access key for the user 
3. Add user to a group which has policies attached

Add users to groups
instead of attaching policies to users.

Adding a User to a Group

1. Create the **IAM** service object
2. Create a **addUserToGroupRequest** object
3. Populate the attributes of the **addUserToGroupRequest** object
4. Tell the IAM service object to **addUserToGroup(addUserToGroupRequest)**
5. Get a **addUserToGroupResult** object back

```
addUserToGroupRequest = CreateObject('java',
'com.amazonaws.services.identitymanagement.model.AddUserToGroupRequest')  2

        .withGroupName(groupName)  3

        .withUserName(userName);

addUserToGroupResult = iam.addUserToGroup(addUserToGroupRequest);  4  5
```

The User Creation Process

1. Create the user 
2. Create the access key for the user 
3. Add user to a group which has policies attached 

The User Creation Process

1. Create the user 
2. Create the access key for the user 
3. Add user to a group which has policies attached 
4. Rotate access keys every [n] days

Use the createdOn property of an access key to determine when to rotate a specific set of keys.

Rotating Access Keys

1. Delete or update the existing access keys
2. If update, set the current keys to “inactive”
3. Make new keys with a **createAccessKeyRequest**

```
deleteAccessKeyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.DeleteAccessKeyRequest')
        .withUserName(userName)
        .withAccessKeyID(userAccessKeyID);
deleteAccessKey = iam.deleteAccessKey(deleteAccessKeyRequest);  1

createAccessKeyRequest = CreateObject('java', 'com.amazonaws.services.identitymanagement.model.CreateAccessKeyRequest')
        .withUserName(application.awsResources.iam.PlayboxUserName);
createAccessKeyResult = iam.createAccessKey(createAccessKeyRequest);  3
accesKeyInfo = createAccessKeyResult.getAccessKey();

userAccessKey = accesKeyInfo.getAccessKeyID();
userSecretKey = accesKeyInfo.getSecretAccessKey();
userKeyCreatedOn = accesKeyInfo.getCreateDate();
```

You can now manage users and permissions in AWS from your CFML app.

Demo time!

Resources Are Not Limitless

- 1500 policies per account
- 300 groups per account
- 10 policies attached to any given user
- 50 tags per resource

HTTPS://DOCS.AWS.AMAZON.COM/GENERAL/LATEST/GR/AWS_SERVICE_LIMITS.HTML

Go Do!

AWS Playbox
https://github.com/brianklaas/awsPlaybox

Using the AWS Java SDK in CFML
https://brianklaas.net/

brian.klaas@gmail.com
@brian_klaas