30 November 2022
Julian Wiegmann

Identity Crisis

Multi Cloud IAM

$ whoami - Julian Wiegmann

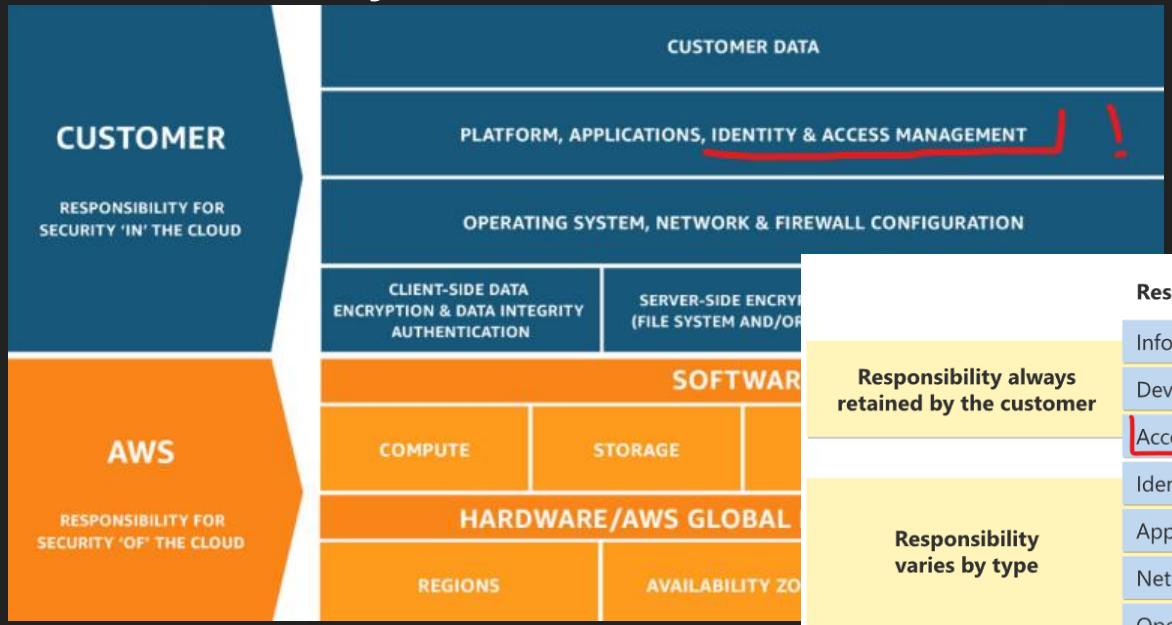
- if worktime < 7:
  - networking, firewalls, solaris/nix*, web proxies, DNS, waf, network intrusion detection
- elif worktime > 7 and worktime < 15:
  - SOC, I&R, SIEM, EDR, detect & prevent, projects like email security, sandboxes, etc.
  - Managing a great team and managing security *implementation and operations* projects
- if oldandwise ?= true:
  - Cloud Security full time

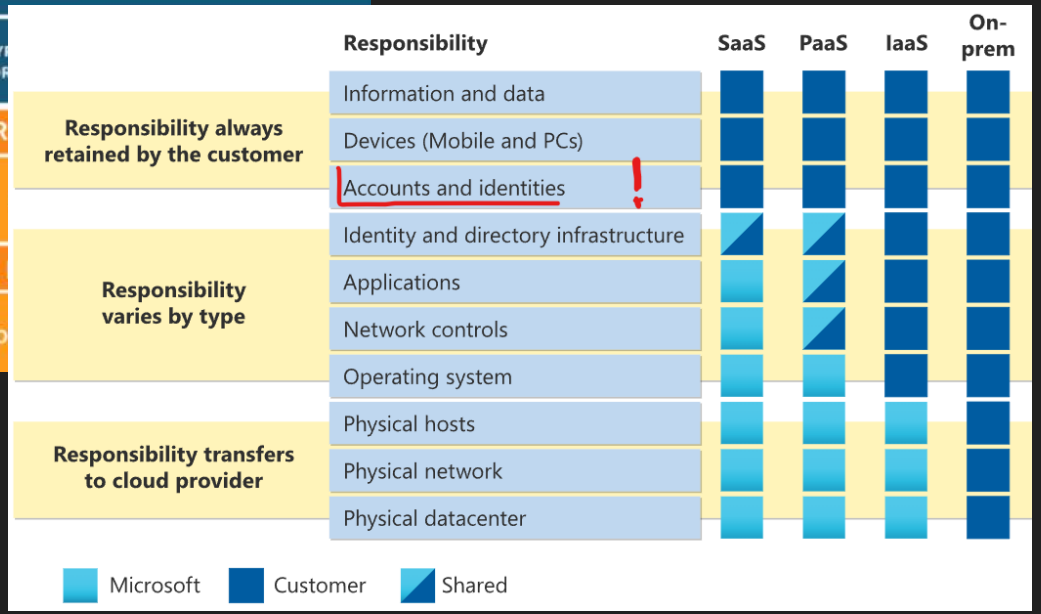
Love cyber security, love learning & challenge of securing companies

Intro

Multi Cloud Security is Challenging
IAM is key to understand

IAM is always relevant





Basics

What is IAM?

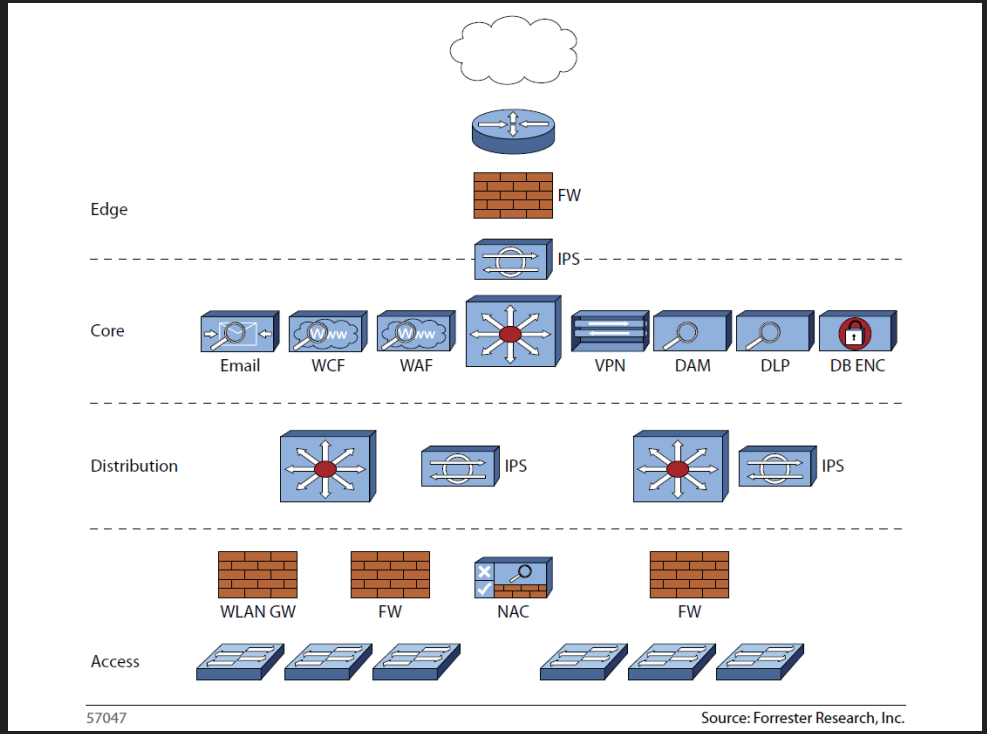

PROCESSES
EVERYWHERE
imgflip.com


HAWA
NUMBER 01-47-87441
DOB 06/03/1981 EXP 06/03/2008
HT WT HAIR EYES SEX CTY
5-10 150 BRO BRO M 0
ISSUE DATE CLASS RESTR ENDORSE
06/18/1998 3
McLOVIN
892 MOMONA ST
HONOLULU, HI 96820


ACCESS
DENIED
imgflip.com

Old School Security

- Bad is on the outside (Internet)
- Secure the perimeter
  - Firewall / DMZ
- Flat and “secure” LAN
- Approach moved to inside LAN
  - Control inside with ‘firewalls’ and vlans etc.
  - *“what can communicate with what”*
- Did not and does not work!

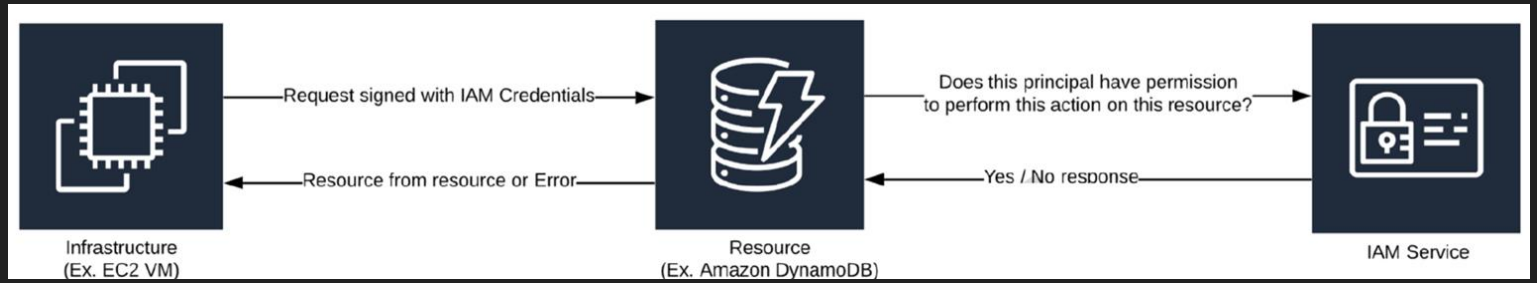


Identity is the new Perimeter

- Cloud is inherently “on the Internet”
- How we work, we want to work and deliver software is
  - “on the Internet / Web”
- Loosely coupled software architectures need to communicate securely in insecure networks
- Everything ‘authS’ and everything has an Identity
- *“who can communicate with who”*

Cloudy IAM

Cloud IAM



- Each cloud has its own IAM (solutionS) and two basic IAM scopes
  - Control Plane
  - Data Plane
- Cloud providers design and build their services ‘around’ their IAM
- Typically two types of identities
  - Humans & "Infrastructure / Apps / Service" identities
- Granular role-based access control
- “Least privilege” & “Zero Trust” is implementable
- *“who can communicate with who”* with granular *“with which permissions” & sometimes “conditions”*

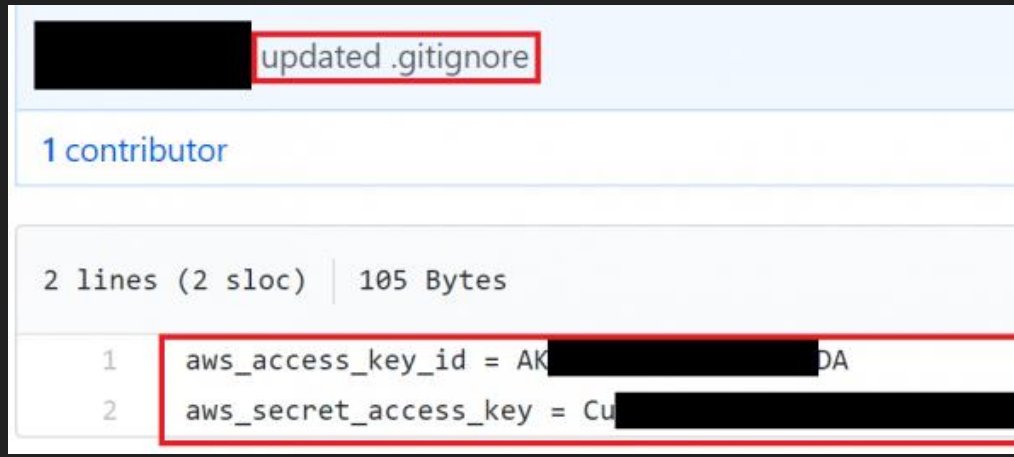
Cloud is Secure

Easy job for me?

No

Biggest threat in cloud security is:

- Misconfiguration (our fault not CSP)
- ***61% of cloud breaches are due to credentials/access***
- Impact of Incident depends on how well you implemented IAM
- Loads of offensive tools for cloud IAM exists (misconfigurations / features) to find and abuse misconfigurations
- Some bad defaults by CSPs around IAM



Study and Crisis

Need to deep dive / learn IAM

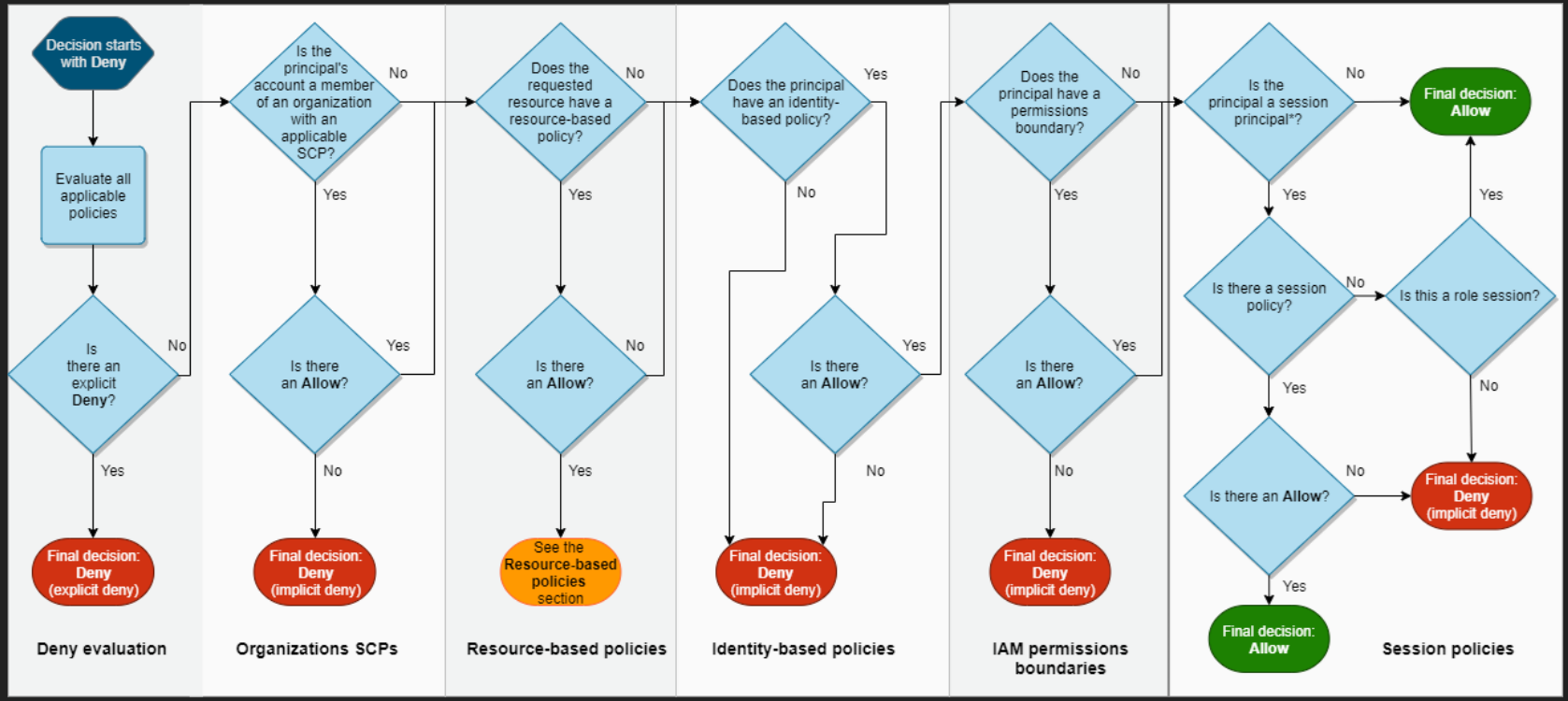
- IAM is king, IAM is key, everything is around IAM
- Of course I get and know IAM generally

But:

- Primary cloud knowledge = GCP
- Basic understanding of Azure and Azure AD too
- Now also need to understand AWS

and I want to really ‘understand’!

Lets understand AWS Policy evaluation logic



*A session principal is either a role session or an IAM federated user session.

But…

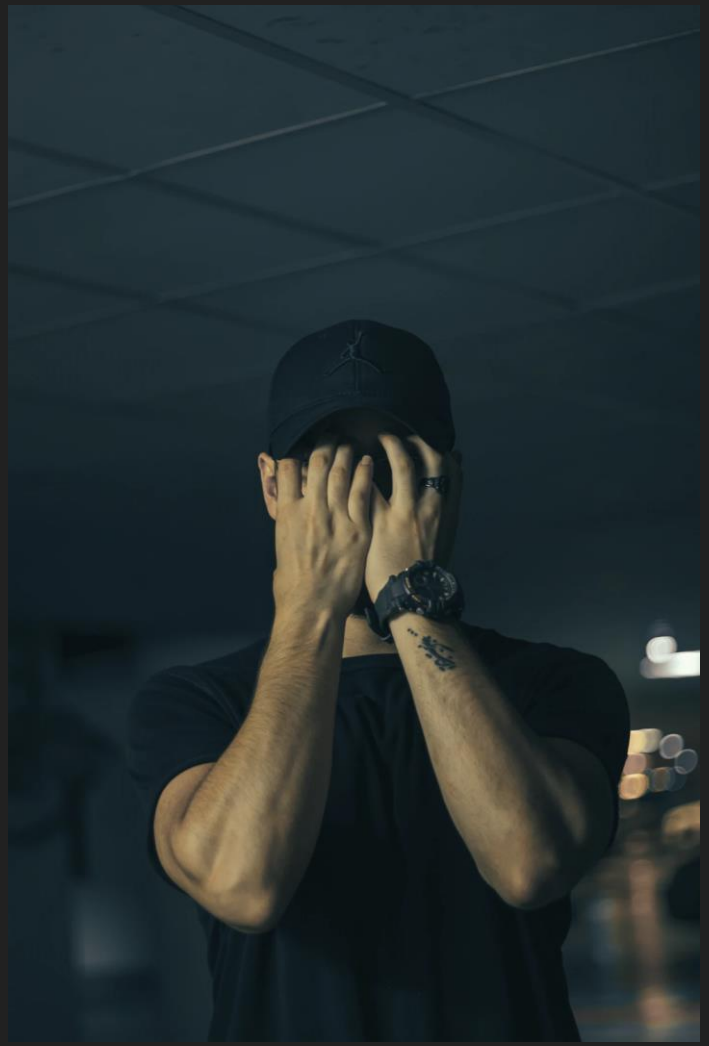
Policies is the hardening / baseline for the cloud control plane & service in Azure

Policies are “Conditional Access Policies” in Azure Active Directory which check “if/when you can authenticate”

IAM Policies in GCP define ‘who’ can do ‘what’ depending on the role that is attached to the resource

Why are there so many steps and different ‘policies’ in AWS…

What did I do…



Understand how “Deny” authorizations works

- Not generally available in GCP, *“transitive”* allow policy system
- Not possible in Azure unless you use Azure Blueprints
- But you can have ‘notActions’ (not allowed?) in Azure “Role Definitions”
- There are implicit denies in AWS “permission boundary”, “Organizations SCPs” or “session policies”
- But also explicit denies in the AWS “IAM policies”

…

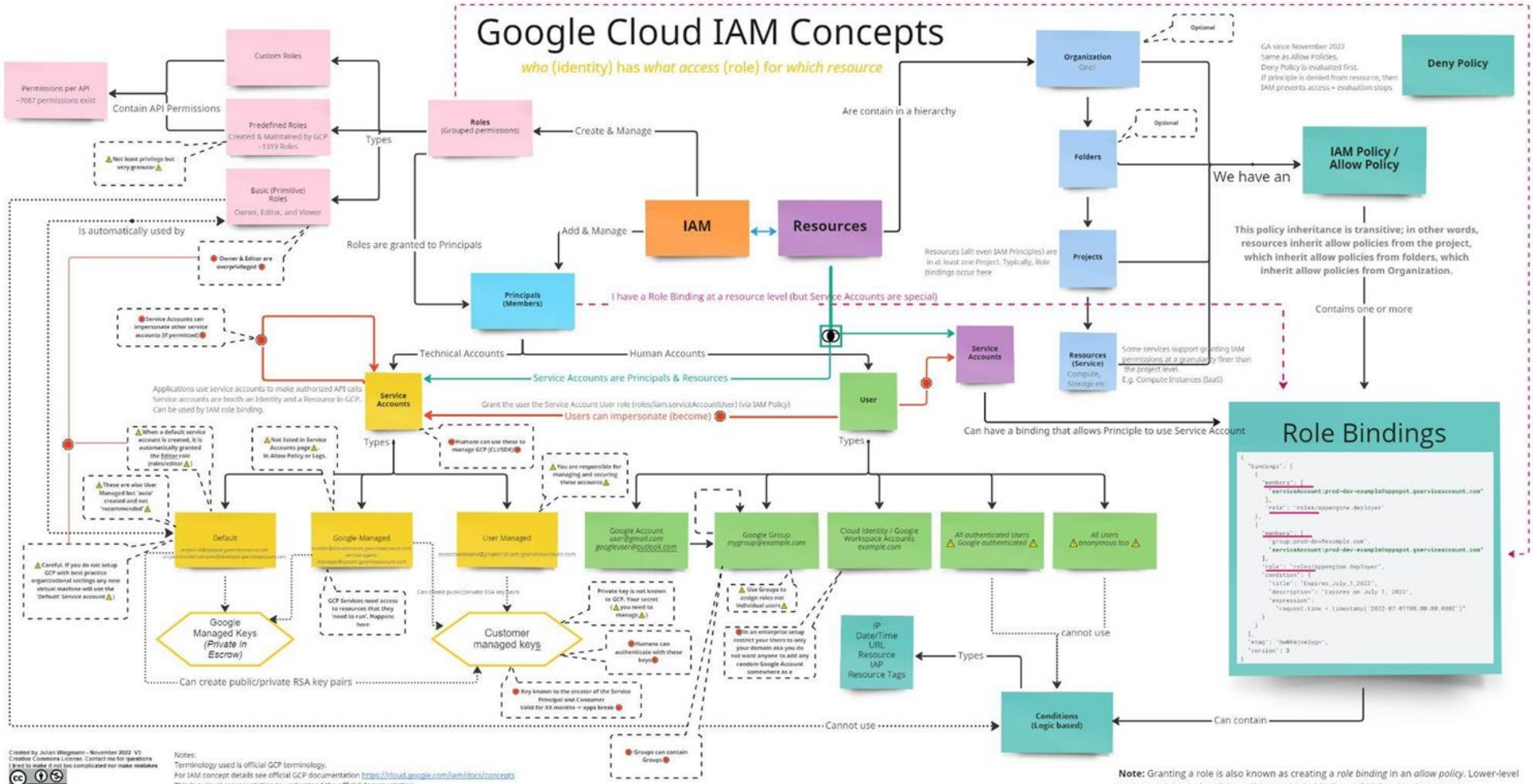
Crisis

Approach

- Slow down
- Focus on one (cloud + topic)
- Make notes on
  - key concepts
  - key terminology
- Mind-map / draw how things relate
- Test / try everything in each cloud

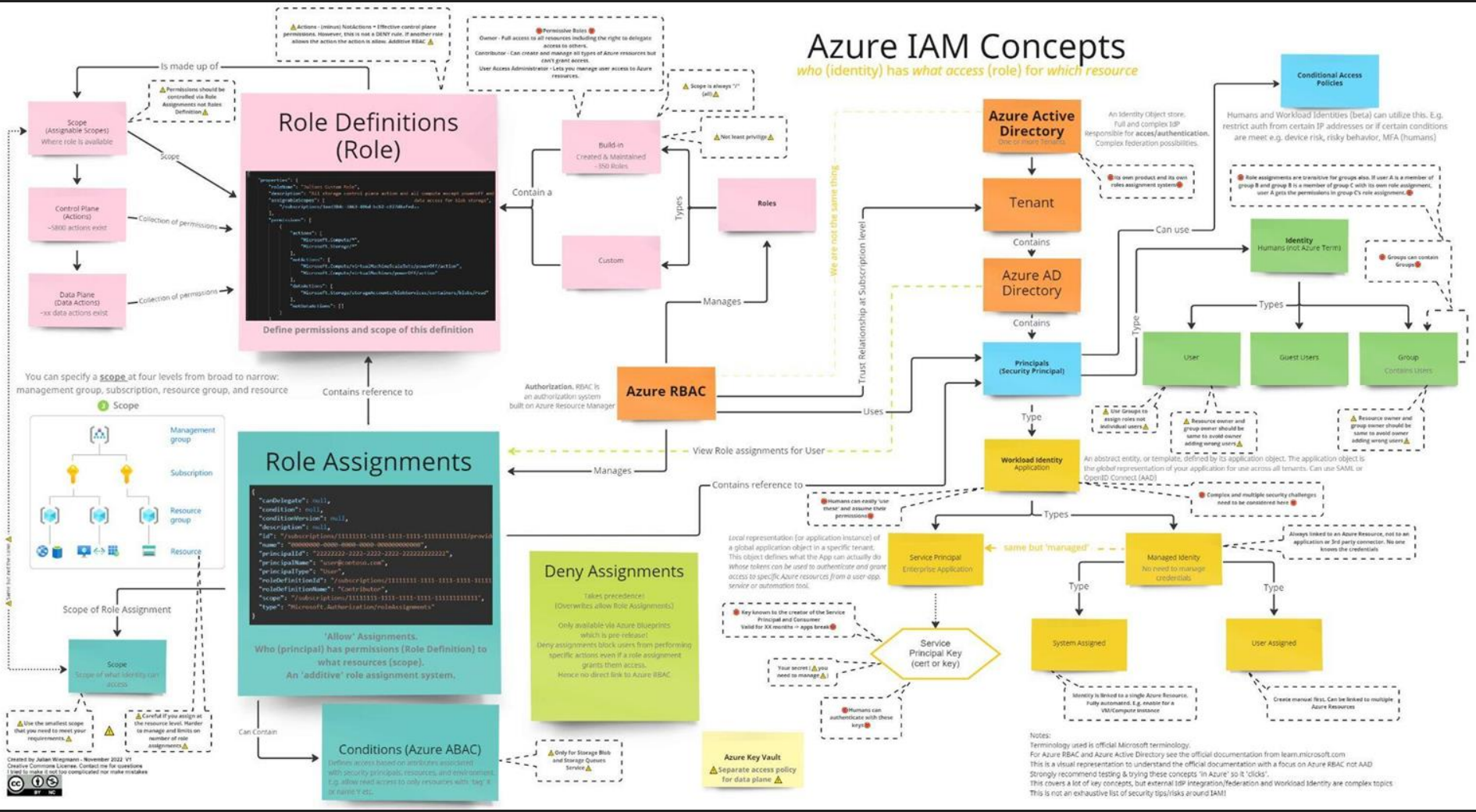
Result


IT IS A CRIME


Google Cloud IAM Concepts
who (identity) has what access (role) for which resource
Permissions per API
~7087 permissions exist
Contain API Permissions
Custom Roles
Predefined Roles
Created & Maintained by GCP
~1319 Roles
Basic (Primitive) Roles
Owner, Editor, and Viewer
Types
Roles
(Grouped permissions)
Create & Manage
Not least privilege but very granular
Is automatically used by
Owner & Editor are overprivileged
Roles are granted to Principals
Add & Manage
IAM
Resources
Are contain in a hierarchy
Organization
Only!
Optional
Folders
Optional
Projects
Resources (all! even IAM Principles) are in at least one Project. Typically, Role bindings occur here
We have an
IAM Policy / Allow Policy
This policy inheritance is transitive; in other words, resources inherit allow policies from the project, which inherit allow policies from folders, which inherit allow policies from Organization.
Contains one or more
GA since November 2022
Same as Allow Policies.
Deny Policy is evaluated first.
If principle is denied from resource, then IAM prevents access + evaluation stops
Deny Policy
Principals
(Members)
I have a Role Binding at a resource level (but Service Accounts are special)
Service Accounts can impersonate other service accounts (if permitted)
Technical Accounts
Human Accounts
Service Accounts are Principals & Resources
Service Accounts
Applications use service accounts to make authorized API calls
Service accounts are both an Identity and a Resource in GCP.
Can be used by IAM role binding.
Grant the user the Service Account User role (roles/iam.serviceAccountUser) (via IAM Policy)
Users can impersonate (become)
User
Service Accounts
Resources (Service)
Compute, Storage etc
Some services support granting IAM permissions at a granularity finer than the project level.
E.g. Compute Instances (IaaS)
Can have a binding that allows Principle to use Service Account
Role Bindings
{
"bindings": [
{
"members": [
"serviceAccount:prod-dev-example@appspot.gserviceaccount.com"
],
"role": "roles/appengine.deployer"
},
{
"members": [
"group:prod-dev@example.com",
"serviceAccount:prod-dev-example@appspot.gserviceaccount.com"
],
"role": "roles/appengine.deployer",
"condition": {
"title": "Expires_July_1_2022",
"description": "Expires on July 1, 2022",
"expression":
"request.time < timestamp('2022-07-01T00:00:00.000Z')"
}
}
],
"etag": "BwWkzvleUgo=",
"version": 3
}
When a default service account is created, it is automatically granted the Editor role (roles/editor)
These are also User Managed but 'auto' created and not 'recommended'
Careful. If you do not setup GCP with best practice organizational settings any new virtual machine will use the 'Default' Service account
Default
Not listed in Service Accounts page
In Allow Policy or Logs.
Google-Managed
GCP Services need access to resources that they 'need to run'. Happens here
Humans can use these to manage GCP (CLI/SDK)
You are responsible for managing and securing these accounts
User Managed
Can create public/private RSA key pairs
Google Managed Keys
(Private in Escrow)
Customer managed keys
Private key is not known to GCP. Your secret (you need to manage)
Humans can authenticate with these keys
Key known to the creator of the Service Principal and Consumer
Valid for XX months -> apps break
Google Account
user@gmail.com
googleuser@outlook.com
Google Group
mygroup@example.com
Use Groups to assign roles not individual users
Cloud Identity / Google Workspace Accounts
example.com
All authenticated Users
Google authenticated
All Users
anonymous too
In an enterprise setup restrict your Users to only your domain aka you do not want anyone to add any random Google Account somewhere as a
IP
Date/Time
URL
Resource
IAP
Resource Tags
Types
cannot use
Conditions
(Logic based)
Can contain
Cannot use
Groups can contain Groups
Created by Julian Wegmann - November 2022 V1
Creative Commons License. Contact me for questions
I tried to make it not too complicated nor make mistakes
Notes:
Terminology used is official GCP terminology.
For IAM concept details see official GCP documentation https://cloud.google.com/iam/docs/concepts
This is a visual representation to understand the official documentation.
Strongly recommend testing & trying these concepts 'in GCP' so it 'clicks'.
This covers a lot of key concepts, but external IdP integration/federation etc. are not covered nor Google Identity / Google Workspace for enterprise usage!
This is not an exhaustive list of security tips/risks around IAM!
Note: Granting a role is also known as creating a role binding in an allow policy. Lower-level resources inherit the allow policies and role bindings of higher-level resources.

Azure IAM Concepts

who (identity) has *what access* (role) for *which resource*



AWS IAM Concepts

who (identity) has *what access* (role) for *which resource*

Created by Julian Wiegmann - November 2022 V1
Creative Commons License. Contact me for questions
I tried to make it not too complicated nor make mistakes

Notes:
Terminology used is official AWS terminology.
For IAM concept details see official AWS documentation
This is not an exhaustive list of security tips/risks around IAM
Nor covers all Identity features/services in AWS

Organization (Optional)
AWS Organizations helps you centrally govern your environment as you scale your workloads in AWS
Can use AWS Organizations service control policy (SCP) to limit the permissions of root user
SCP
Manage & Use
SCP - Service Control Policy
Organization policy to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization.
Contains & Manage
Security credentials are account-specific
AWS Account
AWS account is a container for your AWS resources. You create / manage your AWS resources in an AWS account, & provides administrative capabilities for access and billing
Full Control over
Root can close account
Contains
Manage
Resources (Service)
EC2, S3 etc.
Can use
Access control lists (ACLs)
Cannot be used to control access for a principal within the same account.
Session Policies (for Roles)
No more space but AWS best practice when using federation
Create distinctive role session permissions or to further restrict session permissions, users or systems can set a session policy when assuming a role on the fly.
Session policy are inline permissions policy which users pass, or your identity-broker, when they assume the role.
The effective permissions of the session are the intersection of the role's identity-based policies and the session policy. You can pass a single inline session policy programmatically by using the policy parameter with the AssumeRole, AssumeRoleWithSAML, AssumeRoleWithWebIdentity, and GetFederationToken API operations.
Authorization 2
Similar to resource-based policies, although they do not use the JSON format. Amazon S3, AWS WAF, and Amazon VPC key services that support ACLs.
Signed (authenticated) or unsigned (anonymous) you can configure and not just read. Full control & write to a storage bucket for example are possible, depends on service
Not the same
Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship.
Some service. You can specify ANY user or even whole other AWS accounts to access /edit/ use resources here! This is not a Role
Do not use for day2day tasks
Don't create long-term access keys for your root user. If malicious user gains access to your root user access keys = game over
Root User
Email & Password
Cannot use
Manages
AWS IAM
Not a regional service
Contains
IAM Resources
Root is also a Principal
Roles are both a Principals & Resources
Manage
Policies are attached to Principlas
Authorization
Policies (Permission Policy)
All JSON policy
Type
Resource Based Policy
For Services/Resources
You specify who has access to the resource and what actions they can perform on the resource
Person or application
Principals include federated users and assumed roles
Principals
Sometimes called Identities
Type
Type
Authentication happens here
IAM Entities / Identities (User or Role)
Not Policies managed in Policies but under Roles but "same"
Permission Policy
What can Role do
Trust Policy
Principal (who can assume role)
You can specify ANY user or even whole other AWS accounts to access resources here!
Special Type of Policies
Type
Not recommended by AWS for Users
Are always of type
Control what actions the identity can perform, on which resources, and under what conditions
Identity Based Policies
Attached to an Identity
Type
Inline Policies
Policies that are embedded directly into a single user, group, or role OR Resource
Type
Policies that are attached to a single user, group, or role
Attached here for Roles
Unless Inline
Managed Policies
You can specify who has access to the resource and what actions they can perform on it
Support
Permission Boundaries
A policy that sets the maximum permissions an IAM entities can have
Types
AWS Managed
~998 Policies Exist
Customer Managed
You define
Careful with "FullAccess" Policies
AWS managed policies don't grant least privilege permissions. (this is what AWS says in their documentation. Take note other CSPs
For Users or Roles, NOT Groups!
Group
User Groups
Use Groups to assign permissions not individual users
Contains
User
Account ID / alias user name
Doesn't have to represent an actual person; you can create an IAM user in order to generate an access key for an application that runs in your corporate network and needs AWS access
Requires Policy
AssumeRole
Role
A role is intended to be assumable by anyone who needs it
Cross Account and Cross Service is possible and Confused Deputy Problem
You can allow Roles to assume other Roles. Role Chaining
Subtypes
AWS Service Role
For AWS services to use to do their function
EC2 Service Role
For apps running on EC2
AWS Service Linked Role
predefined by the service include all the permissions
Federation
Oauth, SAML 2.0 (external IDP)
Have their own ARN e.g. arn:aws:iam::aws:policy/IAMReadOnlyAccess
Humans can authenticate with these keys
API based access using access keys
AKIA
Access Keys
Access key ID + Secret Access Key
ASIA
Access Keys
Access key ID + Secret Access Key + expire time
These temporary credentials are called via AWS STS (right?) AssumeRole
Often abused with IMDS v1 SSRF. E.g. vuln webapp can call role creds and attacker can use
Key known to the creator of the key and consumer (if used by 3rd party). Do not expire
Max two access keys per user (for rotation)
Humans can authenticate with these keys

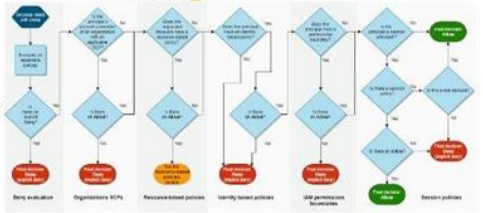
IAM JSON Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "iam:DeleteInstanceProfile",
        "iam:UpdateAssumeRolePolicy",
        "iam:SetDefaultPolicyVersion",
        "iam:ListInstanceProfilesForRole",
        "iam:DetachRolePolicy",
        "iam:CreatePolicyVersion",
        "iam:ListRoleTags",
        "iam:PutRolePermissionsBoundary",
        "iam:ListPolicyVersions",
        "iam:DeleteRole",
        "iam:AttachRolePolicy",
        "iam:ListInstanceProfiles",
        "iam:DeleteRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListRolePolicies",
        "iam:CreateRole",
        "iam:UpdateRole",
        "iam:ListEntitiesForPolicy",
        "iam:DeletePolicy",
        "iam:CreatePolicy",
        "iam:GetRole",
        "iam:ListRoles",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::093525570695:group/*",
        "arn:aws:iam::093525570695:user/*",
        "arn:aws:iam::093525570695:instance-profile/ansible-*",
        "arn:aws:iam::093525570695:policy/*",
        "arn:aws:iam::093525570695:role/*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "194.25.1.1"
        },
        "DateLessThan": {
          "aws:CurrentTime": "2025-06-30T23:59:59Z"
        }
      }
    }
  ]
}
```

Used throughout AWS IAM policies.
Over ~13 000 permissions (actions) exist for the services. Action is allow statement but NotAction (deny) also exist. This can be combined with 'Effect' Deny. E.g. Deny IAM* except Multifactor was done.
Scope can be defined on resources.
Wildcards are possible throughout.
Conditions can use Boolean expressions.
Can include things like "must MFA before allow"

Policy evaluation logic
This is key

If a single permissions policy includes a denied action, AWS denies the entire request and stops evaluating. This is called an *explicit deny*. Because requests *are denied by default*



Free to use

Medium: Visualizing Multi Cloud IAM Concepts

Short: shorturl.at/ceorT

Some tips around IAM

- Take it slow, try and test in each cloud what you learned step-by-step
- You cannot defend it if you do not know how the attackers hack it (basics knowledge is enough)
  - Always use ATT&CK, pentesting, red teaming talks/videos/github tools and knowledge sharing to understand how IAM can be hacked/abused/used by malicious actor

Thank you