30 November 2022
Julian Wiegmann

Identity Crisis

Multi Cloud IAM

$ whoami - Julian Wiegmann

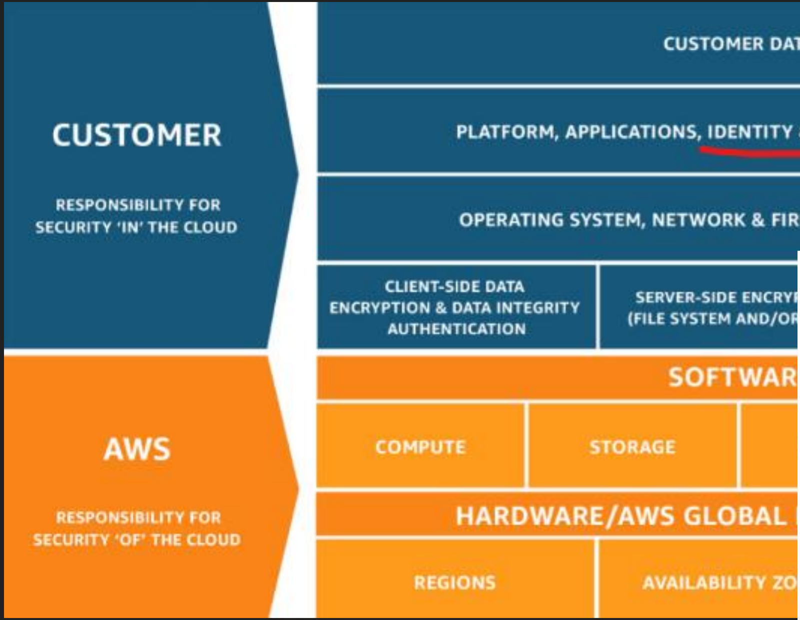
- if worktime < 7:
  - networking, firewalls, solaris/nix*, web proxies, DNS, waf, network intrusion detection
- elif worktime > 7 and worktime < 15:
  - SOC, I&R, SIEM, EDR, detect & prevent, projects like email security, sandboxes, etc.
  - Managing a great team and managing security *<u>implementation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



CUSTOMER

RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD

CUSTOMER DATA

PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT !

OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION

CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION

SERVER-SIDE ENCRY
(FILE SYSTEM AND/OR

AWS

RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD

SOFTWAR

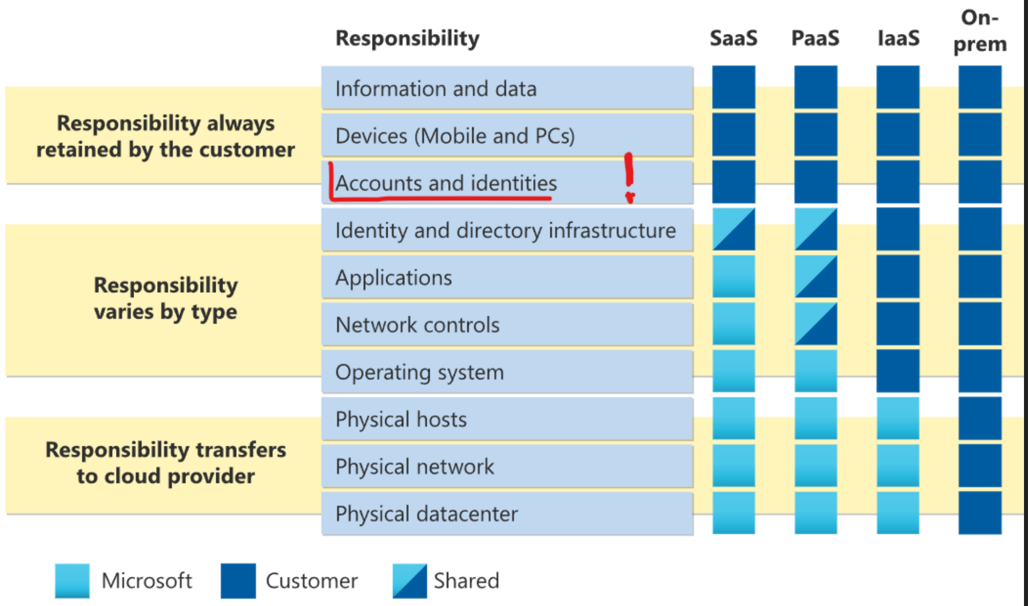
COMPUTE

STORAGE

HARDWARE/AWS GLOBAL I

REGIONS

AVAILABILITY ZO



	Responsibility	SaaS	PaaS	IaaS	On-prem
Responsibility always retained by the customer	Information and data	■	■	■	■
	Devices (Mobile and PCs)	■	■	■	■
	Accounts and identities !	■	■	■	■
Responsibility varies by type	Identity and directory infrastructure	◩	◩	■	■
	Applications	□	◩	■	■
	Network controls	□	◩	■	■
	Operating system	□	□	■	■
Responsibility transfers to cloud provider	Physical hosts	□	□	□	■
	Physical network	□	□	□	■
	Physical datacenter	□	□	□	■

□ Microsoft ■ Customer ◩ Shared

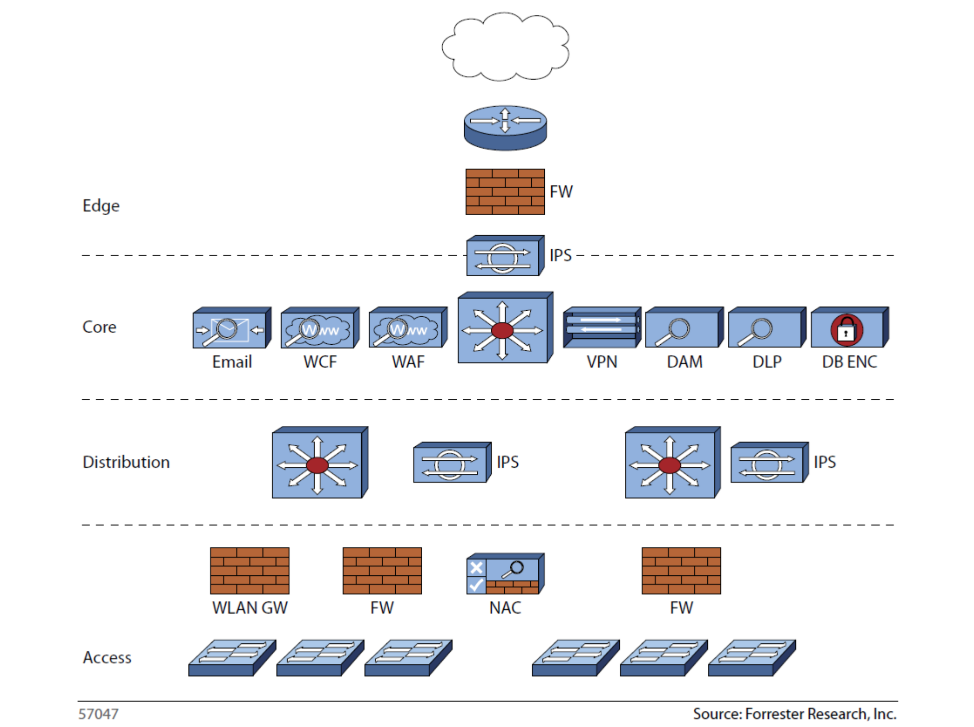
Basics

What is IAM?

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
McLOVIN
892 MOMONA ST
HONOLULU, HI 96820
PROCESSES
EVERYWHERE
imgflip.com
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!



Source: Forrester Research, Inc.

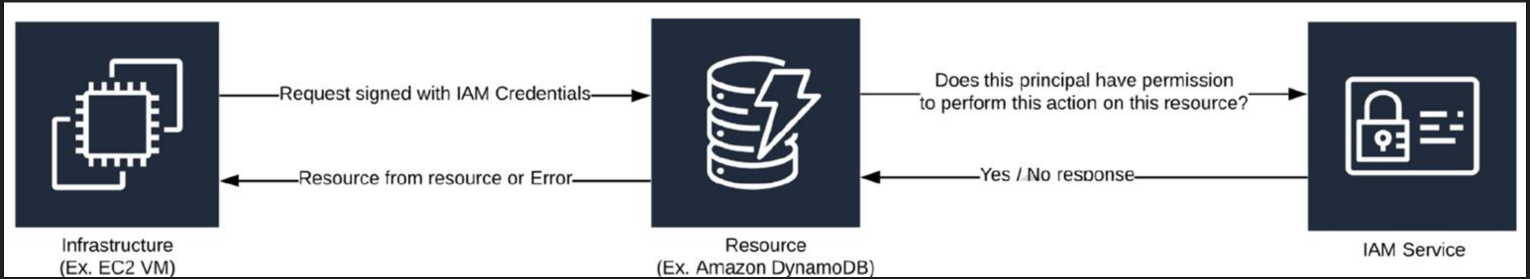
Identity is the new Perimeter

- 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”*

YOU PEOPLE GET AN IDENTITY!
YOU MACHINE GET AN IDENTITY!
YOU APPLICATION GET AN IDENTITY!
EVERYTHING GETS AN IDENTITY!
imgflip.com

Cloudy IAM

Cloud IAM


Request signed with IAM Credentials
Resource from resource or Error
Does this principal have permission to perform this action on this resource?
Yes / No response
Infrastructure (Ex. EC2 VM)
Resource (Ex. Amazon DynamoDB)
IAM Service

- 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”*

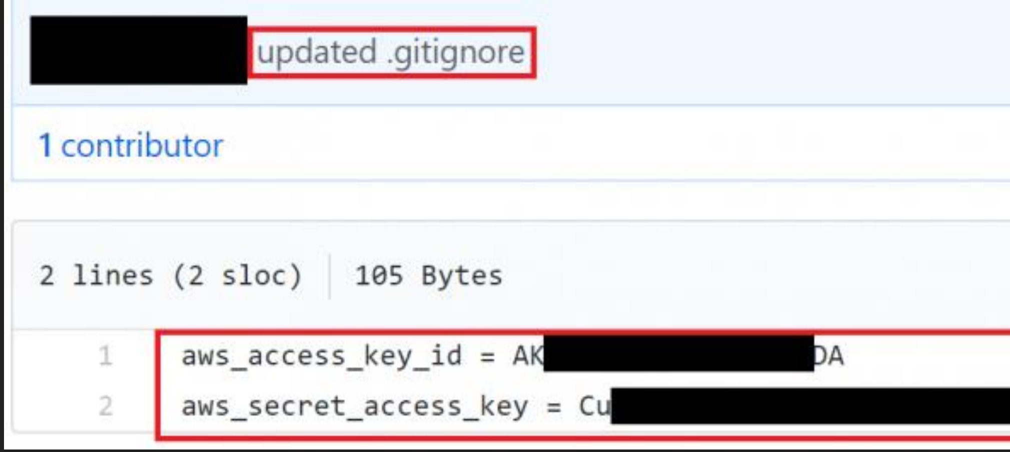
Cloud is Secure

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



[redacted] updated .gitignore

1 contributor

2 lines (2 sloc) | 105 Bytes

```
aws_access_key_id = AK[redacted]DA
aws_secret_access_key = Cu[redacted]
```

Study and Crisis

Need to deep dive / learn IAM

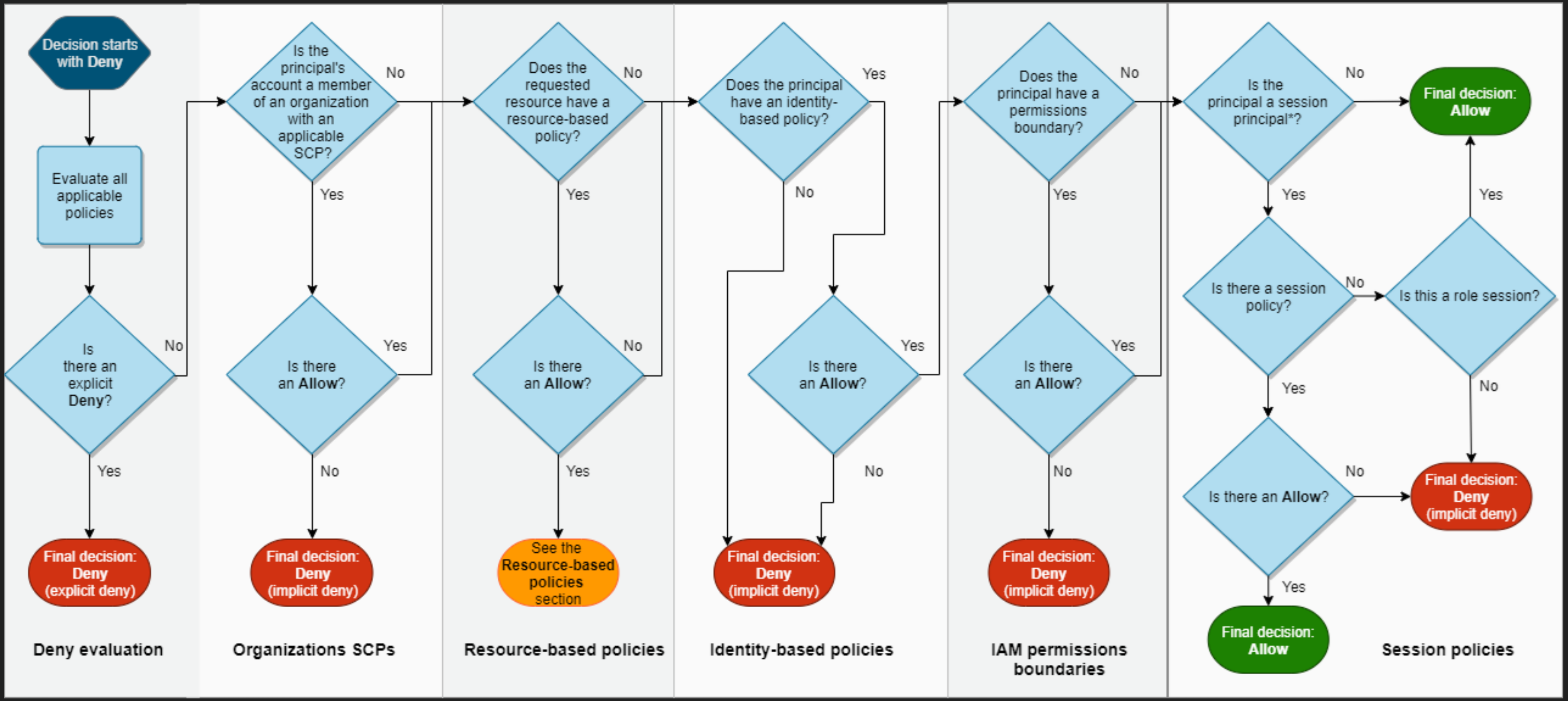
- IAM is king, IAM is key, everything is around IAM
- Of course I get and know IAM generally

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


Decision starts with Deny
Evaluate all applicable policies
Is there an explicit Deny?
No
Yes
Final decision: Deny (explicit deny)
Deny evaluation
Is the principal's account a member of an organization with an applicable SCP?
No
Yes
Is there an Allow?
Yes
No
Final decision: Deny (implicit deny)
Organizations SCPs
Does the requested resource have a resource-based policy?
No
Yes
Is there an Allow?
No
Yes
See the Resource-based policies section
Resource-based policies
Does the principal have an identity-based policy?
Yes
No
Is there an Allow?
Yes
No
Final decision: Deny (implicit deny)
Identity-based policies
Does the principal have a permissions boundary?
No
Yes
Is there an Allow?
Yes
No
Final decision: Deny (implicit deny)
IAM permissions boundaries
Is the principal a session principal*?
No
Final decision: Allow
Yes
Is there a session policy?
No
Is this a role session?
Yes
No
Yes
Is there an Allow?
No
Final decision: Deny (implicit deny)
Yes
Final decision: Allow
Session policies

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

But…

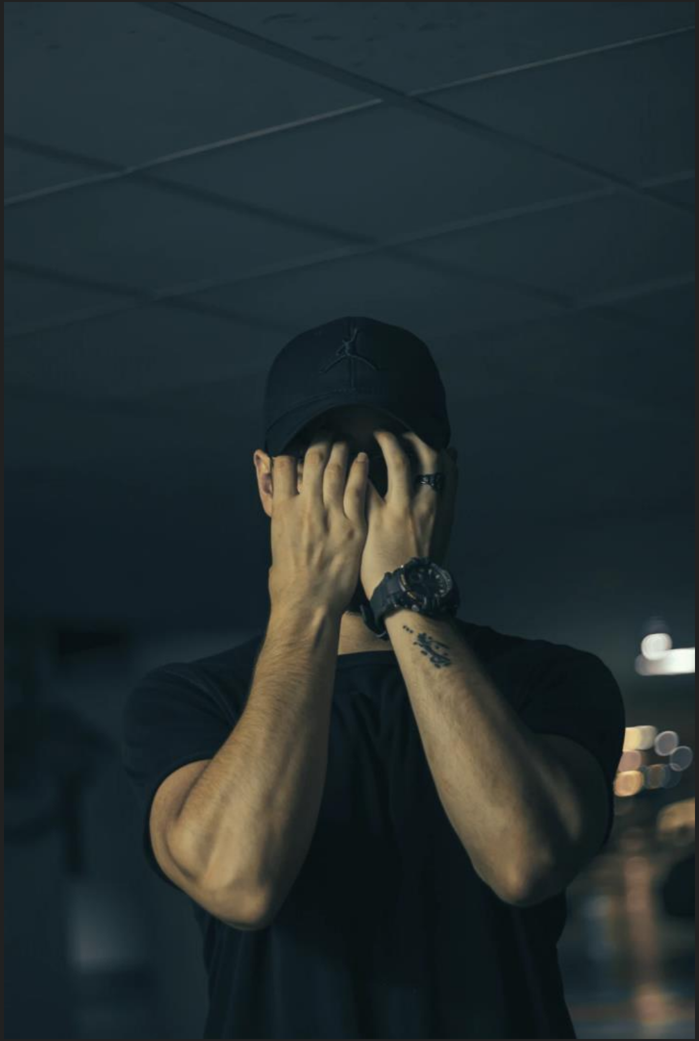
Policies is the hardening / baseline for the cloud control plane & service in Azure

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

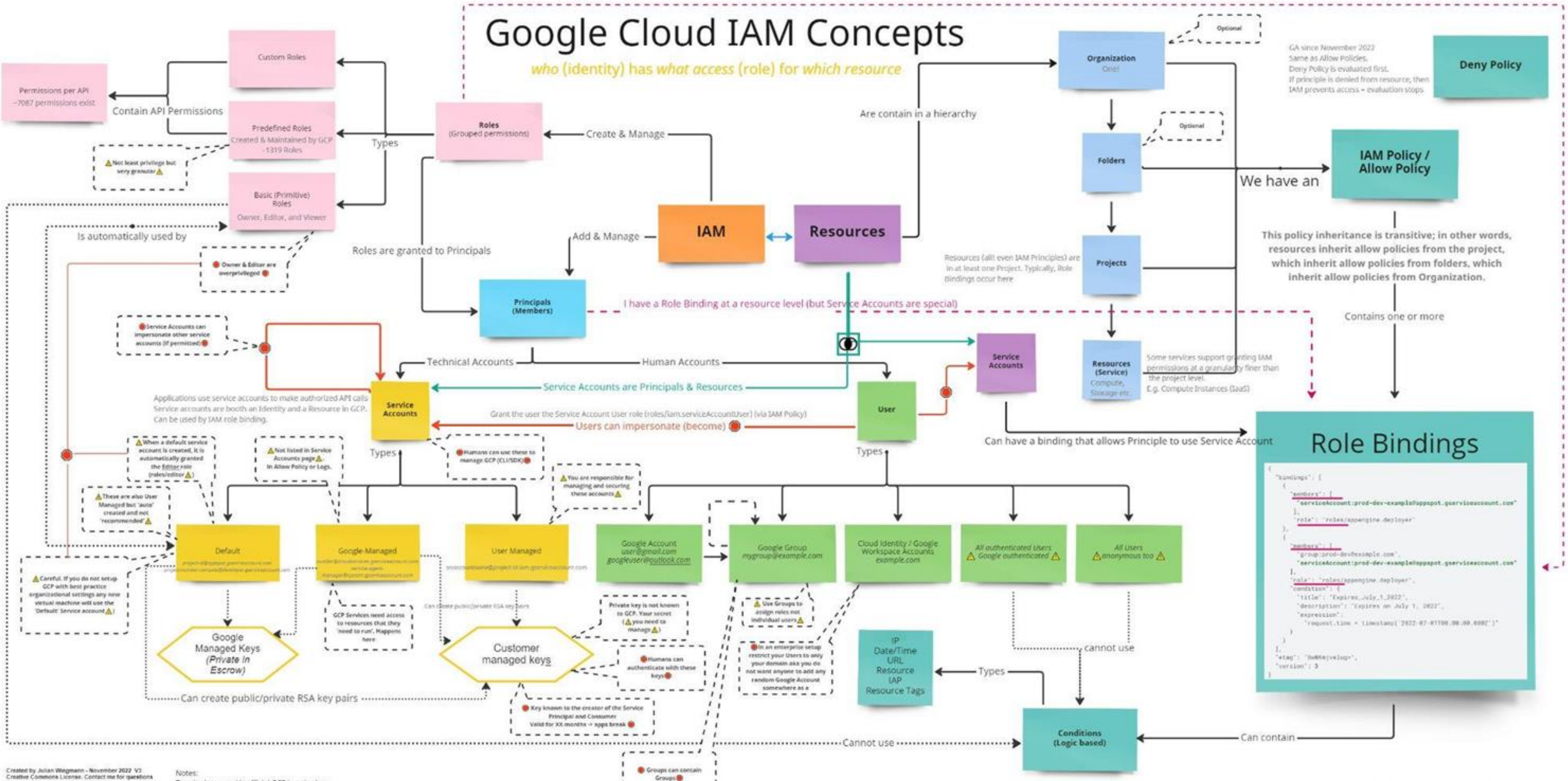
- Slow down
- Focus on one (cloud + topic)
- Make notes on
  - key concepts
  - key terminology
- Mind-map / draw how things relate
- Test / try everything in each cloud

Result

The fish in this pond are being actively farmed
IT IS A CRIME...

Google Cloud IAM Concepts

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



IAM
Resources
Roles
(Grouped permissions)
Create & Manage
Types
Custom Roles
Predefined Roles
Created & Maintained by GCP
~1319 Roles
Basic (Primitive) Roles
Owner, Editor, and Viewer
Permissions per API
~7087 permissions exist
Contain API Permissions
⚠ Not least privilege but very granular ⚠
Is automatically used by
● Owner & Editor are overprivileged ●
Roles are granted to Principals
Add & Manage
Principals
(Members)
I have a Role Binding at a resource level (but Service Accounts are special)
Are contain in a hierarchy
Organization
One!
Optional
Folders
Optional
Projects
Resources (Service)
Compute, Storage etc.
We have an
IAM Policy / Allow Policy
Deny Policy
Contains one or more
Role Bindings
Can contain
Conditions (Logic based)
Types
IP
Date/Time
URL
Resource
IAP
Resource Tags
cannot use
Cannot use
Technical Accounts
Human Accounts
Service Accounts
Service Accounts are Principals & Resources
Grant the user the Service Account User role (roles/iam.serviceAccountUser) (via IAM Policy)
Users can impersonate (become) ●
User
Service Accounts
Can have a binding that allows Principle to use Service Account
● Service Accounts can impersonate other service accounts (if permitted) ●
● Humans can use these to manage GCP (CLI/SDK) ●
Types
Default
project-id@appspot.gserviceaccount.com
project-number-compute@developer.gserviceaccount.com
Google-Managed
number@cloudservices.gserviceaccount.com
service-agent
manager@system.gserviceaccount.com
User Managed
service-account-name@project-id.iam.gserviceaccount.com
⚠ 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 ⚠)
⚠ Not listed in Service Accounts page ⚠. In Allow Policy or Logs.
⚠ You are responsible for managing and securing these accounts ⚠
GCP Services need access to resources that they 'need to run'. Happens here
Google Managed Keys (Private In Escrow)
Customer managed keys
Can create public/private RSA key pairs
Can create public/private RSA key pairs
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 ●
Types
Google Account
user@gmail.com
googleuser@outlook.com
Google Group
mygroup@example.com
Cloud Identity / Google Workspace Accounts
example.com
All authenticated Users
⚠ Google authenticated ⚠
All Users
⚠ anonymous too ⚠
⚠ Use Groups to assign roles not individual users ⚠
● 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
● Groups can contain Groups ●

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

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.

Resources (all! even IAM Principles) are in at least one Project. Typically, Role Bindings occur here

Some services support granting IAM permissions at a granularity finer than the project level.
E.g. Compute Instances (IaaS)

Applications use service accounts to make authorized API calls.
Service accounts are booth an Identity and a Resource in GCP.
Can be used by IAM role binding.

```
{
  "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": "BwWKmjvelug=",
  "version": 3
}
```

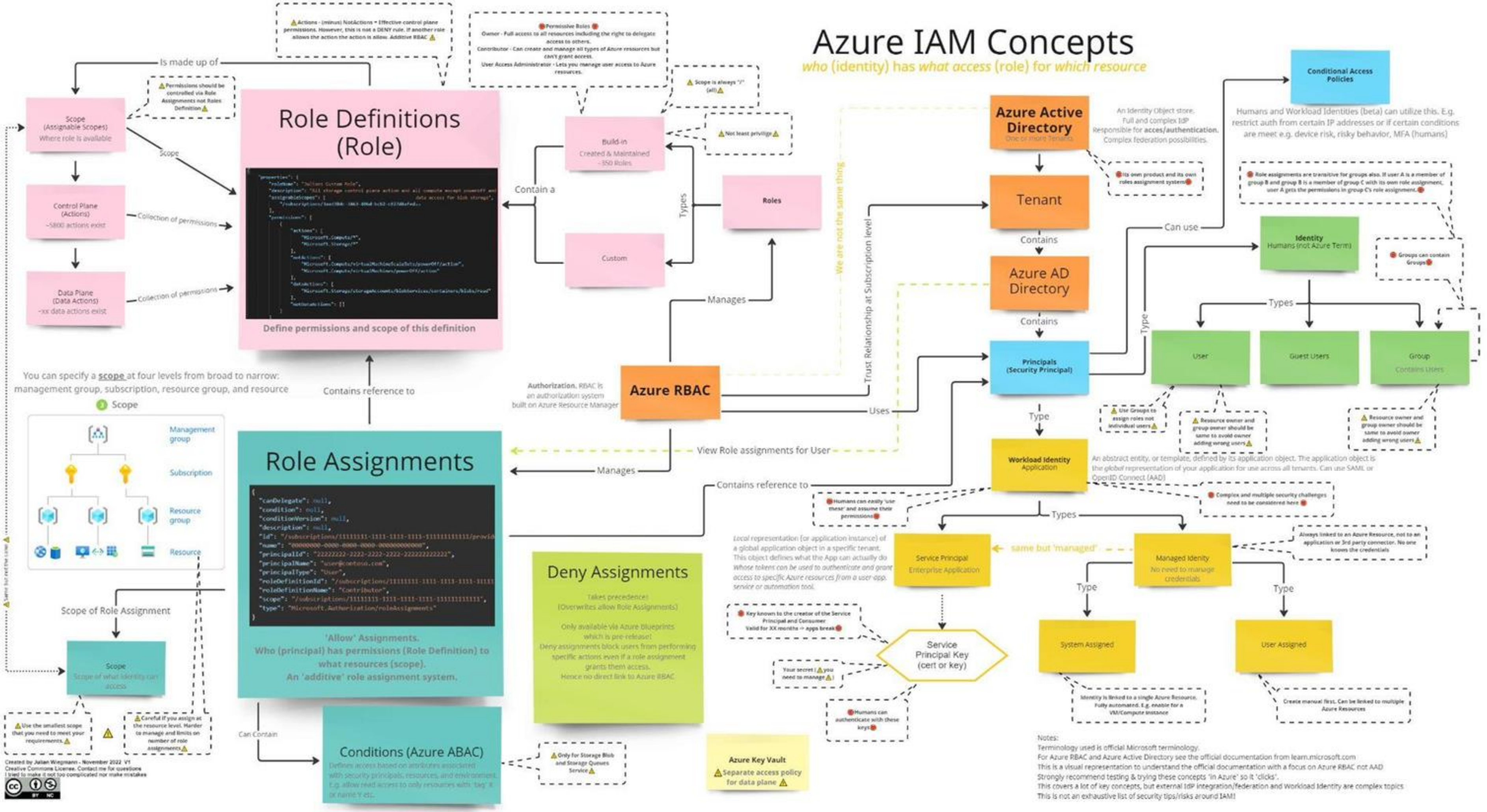
Created by Julian Wiegmann - November 2022 V3
Contact me for questions
I tried to make it not too complicated nor make mistakes

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*



Role Definitions (Role)

```
{
    "properties": {
        "roleName": "Julians Custom Role",
        "description": "All storage control plane action and all compute except poweroff and
                                                    data access for blob storage",
        "assignableScopes": [
            "/subscriptions/[illegible]"
        ],
        "permissions": [
            {
                "actions": [
                    "Microsoft.Compute/*",
                    "Microsoft.Storage/*"
                ],
                "notActions": [
                    "Microsoft.Compute/virtualMachineScaleSets/powerOff/action",
                    "Microsoft.Compute/virtualMachines/powerOff/action"
                ],
                "dataActions": [
                    "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read"
                ],
                "notDataActions": []
            }
```

Define permissions and scope of this definition

⚠ Actions - (minus) NotActions = Effective control plane permissions. However, this is not a DENY rule. If another role allows the action the action is allow. Additive RBAC ⚠

Is made up of

Scope (Assignable Scopes) — Where role is available

⚠ Permissions should be controlled via Role Assignments not Roles Definition ⚠

Scope

Control Plane (Actions) — ~5800 actions exist

Collection of permissions

Data Plane (Data Actions) — ~xx data actions exist

Collection of permissions

🔴 Permissive Roles 🔴
Owner - Full access to all resources including the right to delegate access to others.
Contributor - Can create and manage all types of Azure resources but can't grant access.
User Access Administrator - Lets you manage user access to Azure resources.

⚠ Scope is always "/" (all) ⚠

⚠ Not least privilige ⚠

Build-in — Created & Maintained ~350 Roles

Custom

Contain a

Types

Roles

Manages

You can specify a **scope** at four levels from broad to narrow: management group, subscription, resource group, and resource

⚠ Same but not the same ⚠

Contains reference to

Role Assignments

```
{
  "canDelegate": null,
  "condition": null,
  "conditionVersion": null,
  "description": null,
  "id": "/subscriptions/11111111-1111-1111-1111-111111111111/provid
  "name": "00000000-0000-0000-0000-000000000000",
  "principalId": "22222222-2222-2222-2222-222222222222",
  "principalName": "user@contoso.com",
  "principalType": "User",
  "roleDefinitionId": "/subscriptions/11111111-1111-1111-1111-11111
  "roleDefinitionName": "Contributor",
  "scope": "/subscriptions/11111111-1111-1111-1111-111111111111",
  "type": "Microsoft.Authorization/roleAssignments"
}
```

**'Allow' Assignments.
Who (principal) has permissions (Role Definition) to what resources (scope).
An 'additive' role assignment system.**

Scope of Role Assignment

Scope — Scope of what identity can access

⚠ Use the smallest scope that you need to meet your requirements. ⚠

⚠

⚠ Careful if you assign at the resource level. Harder to manage and limits on number of role assignments ⚠

Can Contain

Conditions (Azure ABAC)

Defines access based on attributes associated with security principals, resources, and environment. E.g. allow read access to only resources with 'tag' X or name Y etc.

⚠ Only for Storage Blob and Storage Queues Service ⚠

Deny Assignments

Takes precedence!
(Overwrites allow Role Assignments)

Only available via Azure blueprints which is pre-release!
Deny assignments block users from performing specific actions even if a role assignment grants them access.
Hence no direct link to Azure RBAC

Azure Key Vault
⚠ Separate access policy for data plane ⚠

Authorization. RBAC is an authorization system built on Azure Resource Manager

Azure RBAC

Manages

View Role assignments for User

Manages

Contains reference to

Uses

Trust Relationship at Subscription level

We are not the same thing

Azure Active Directory — One or more Tenants

An Identity Object store.
Full and complex IdP
Responsible for **acces/authentication.**
Complex federation possibilities.

🔴 Its own product and its own roles assignment system 🔴

Tenant

Contains

Azure AD Directory

Contains

Principals (Security Principal)

Can use

Conditional Access Policies

Humans and Workload Identities (beta) can utilize this. E.g. restrict auth from certain IP addresses or if certain conditions are meet e.g. device risk, risky behavior, MFA (humans)

🔴 Role assignments are transitive for groups also. If user A is a member of group B and group B is a member of group C with its own role assignment, user A gets the permissions in group C's role assignment. 🔴

Type

Identity — Humans (not Azure Term)

🔴 Groups can contain Groups 🔴

Types

User

Guest Users

Group — Contains Users

⚠ Use Groups to assign roles not individual users ⚠

⚠ Resource owner and group owner should be same to avoid owner adding wrong users ⚠

⚠ Resource owner and group owner should be same to avoid owner adding wrong users ⚠

Type

Workload Identity — Application

An abstract entity, or template, defined by its application object. The application object is the *global* representation of your application for use across all tenants. Can use SAML or OpenID Connect (AAD)

🔴 Complex and multiple security challenges need to be considered here 🔴

🔴 Humans can easily 'use these' and assume their permissions 🔴

Types

Local representation (or application instance) of a global application object in a specific tenant.
This object defines what the App can actually do
Whose tokens can be used to authenticate and grant access to specific Azure resources from a user-app, service or automation tool.

Service Principal — Enterprise Application

same but 'managed'

Managed Idenity — No need to manage credentials

Always linked to an Azure Resource, not to an application or 3rd party connector. No one knows the credentials

Service Principal Key (cert or key)

🔴 Key known to the creator of the Service Principal and Consumer
Valid for XX months -> apps break 🔴

Your secret (⚠ you need to manage ⚠)

🔴 Humans can authenticate with these keys 🔴

Type

System Assigned

Identity is linked to a single Azure Resource. Fully automated. E.g. enable for a VM/Compute instance

Type

User Assigned

Create manual first. Can be linked to multiple Azure Resources

Notes:
Terminology used is official Microsoft terminology.
For Azure RBAC and Azure Active Directory see the official documentation from learn.microsoft.com
This is a visual representation to understand the official documentation with a focus on Azure RBAC not AAD
Strongly recommend testing & trying these concepts 'in Azure' so it 'clicks'.
This covers a lot of key concepts, but external IdP integration/federation and Workload Identity are complex topics
This is not an exhaustive list of security tips/risks around IAM!

Created by Julian Wiegmann - November 2022 V1

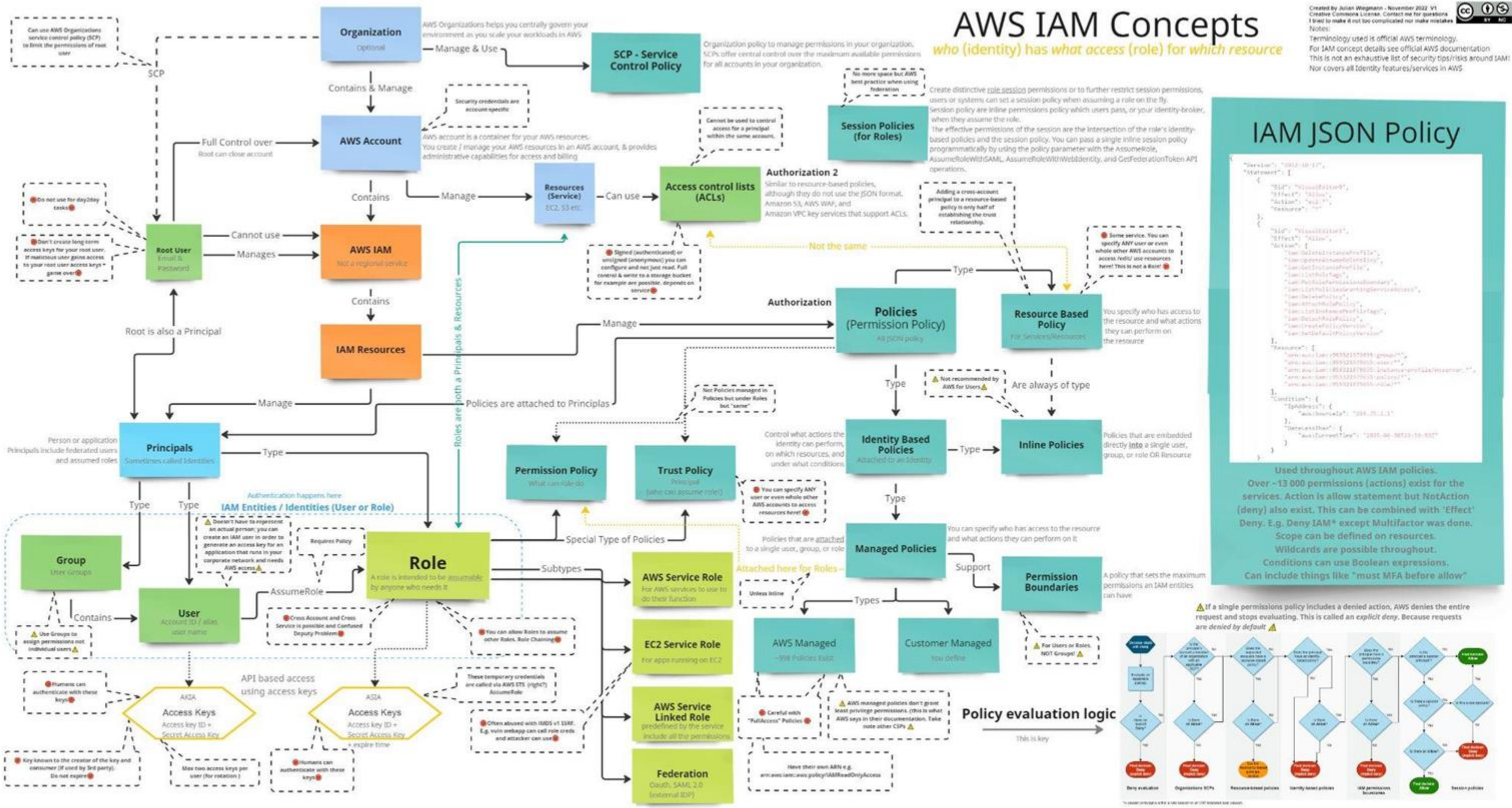
I tried to make it not too complicated nor make mistakes

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



AWS Organizations helps you centrally govern your environment as you scale your workloads in AWS

Organization policy to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization.

Can use AWS Organizations service control policy (SCP) to limit the permissions of root user

Security credentials are account specific

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

● 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 ●

Cannot be used to control access for a principal within the same account.

● 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 ●

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.

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.

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! ●

You specify who has access to the resource and what actions they can perform on the resource

⚠ Not recommended by AWS for Users ⚠

Control what actions the identity can perform, on which resources, and under what conditions

Policies that are embedded directly into a single user, group, or role OR Resource

Policies that are attached to a single user, group, or role

You can specify who has access to the resource and what actions they can perform on it

A policy that sets the maximum permissions an IAM entities can have

⚠ For Users or Roles. NOT Groups! ⚠

Unless Inline

Not Policies managed in Policies but under Roles but "same"

● You can specify ANY user or even whole other AWS accounts to access resources here! ●

Person or application
Principals include federated users and assumed roles

⚠ 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

A role is intended to be assumable by anyone who needs it

⚠ Use Groups to assign permissions not individual users ⚠

● Cross Account and Cross Service is possible and Confused Deputy Problem ●

● You can allow Roles to assume other Roles. Role Chaining ●

● Humans can authenticate with these keys ●

● 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 ●

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 ●

● 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 ⚠

Have their own ARN e.g.
arn:aws:iam::aws:policy/IAMReadOnlyAccess

IAM JSON Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteInstanceProfile",
                "iam:[illegible]",
                "iam:GetInstanceProfile",
                "iam:ListRoleTags",
                "iam:PutRolePermissionsBoundary",
                "iam:ListPoliciesGrantingServiceAccess",
                "iam:DeletePolicy",
                "iam:AttachRolePolicy",
                "iam:ListInstanceProfileTags",
                "iam:DetachRolePolicy",
                "iam:CreatePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": [
                "arn:aws:iam::[illegible]:group/*",
                "arn:aws:iam::[illegible]:user/*",
                "arn:aws:iam::[illegible]:instance-profile/[illegible]*",
                "arn:aws:iam::[illegible]:policy/*",
                "arn:aws:iam::[illegible]:role/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "[illegible]"
                },
                "DateLessThan": {
                    "aws:CurrentTime": "[illegible]"
                }
            }
        }
    }
```

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"

⚠ 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