



# AWS Cloud Security & Access Management

## Key Takeaways

# All rights reserved to nnSoftware GmbH

No part of this publication may be reproduced, copied, transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of nnSoftware GmbH

# About TechWorld with Nana

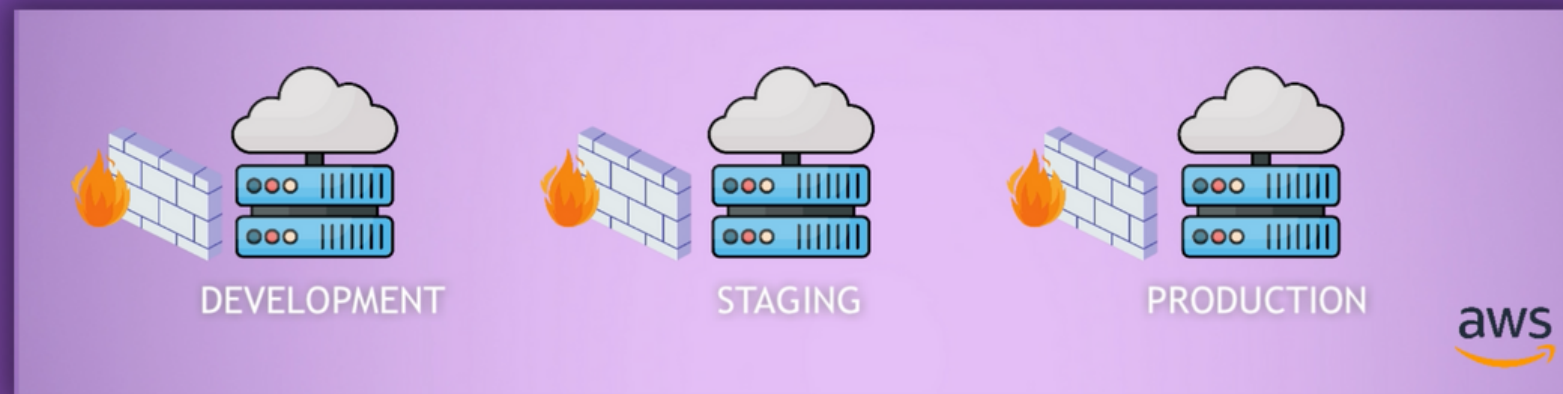
TechWorld with Nana is an established name in the DevOps and Cloud industry, and it stands for the quality trainings helping 1,000s of engineers acquire the most in-demand skills in this field.



Our mission is enable individual engineers as well as companies to take advantage of the recent developments in Cloud and DevOps fields, to use technologies and concepts in order to create efficient, automated, streamlined DevSecOps processes in organisations.

# Securing the Deployment Environment

- We need to **secure the underlying infrastructure**, where our container runs
- This is the server and the whole infrastructure network (in our case AWS infrastructure)



1. We need to not only **secure the network and servers** itself



2. But also the **access to the services** on the cloud platform



**All security efforts are wasted, if you don't have proper access management**

## AWS Account Security:

- Who has access to what resources?
- Users and Permissions

# Administration vs Usage

- Platforms, like cloud platforms, Kubernetes, GitLab CI server etc. always have **2 aspects**

## Administration

- Configure and set up the platform
- Administration has **many possibilities of high impact security misconfigurations**
- Most important part is **Access Management**, managing user access to resources and services on a platform



## Usage

- Use platform for its actual purpose

# AWS Access Management



# Introduction to AWS Access Management

- The AWS Service that allows you to manage access to AWS resources is called IAM
- It's a crucial component of AWS security to protect your AWS resources from unauthorized access

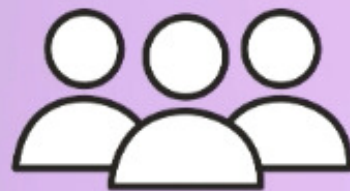
## Identity and Access Management (IAM)



- You do that by creating and configuring IAM users, groups, roles and policies:



IAM User



IAM Groups



IAM Policies



IAM Roles



# Securing AWS Account

- ✓ **Don't use ROOT user.** Instead create own users for different purposes with less permissions
- ✓ **Delete Access Keys for Root User.** Root user should not have programmatic access at all!
- ✓ **Configure MFA** for all AWS accounts

## Root User

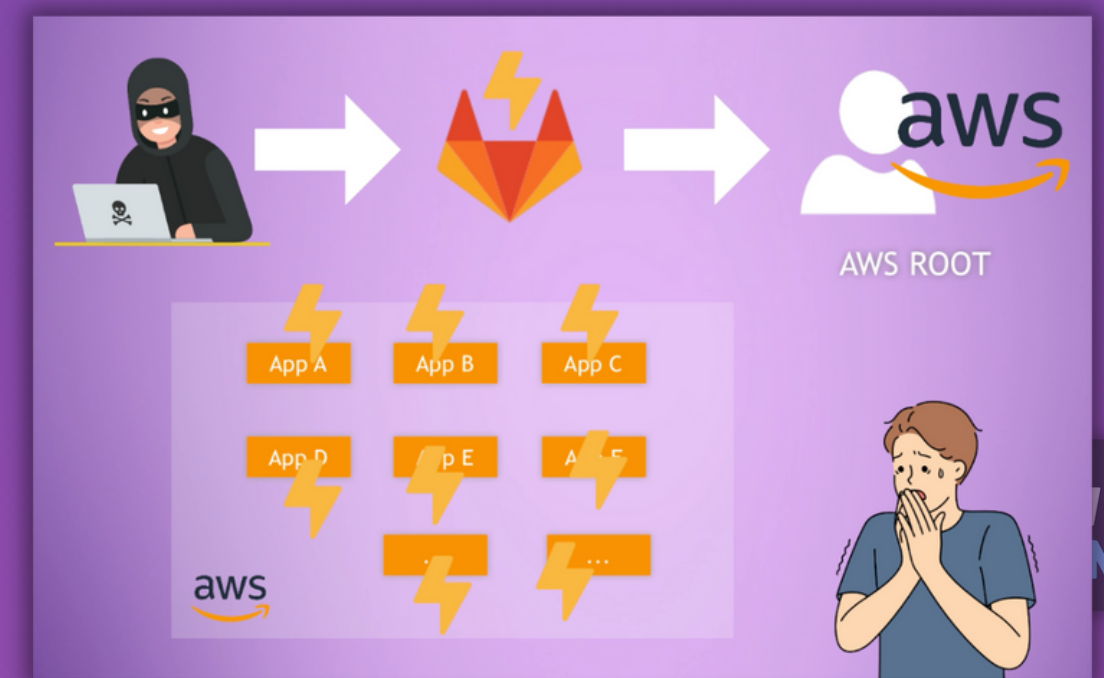
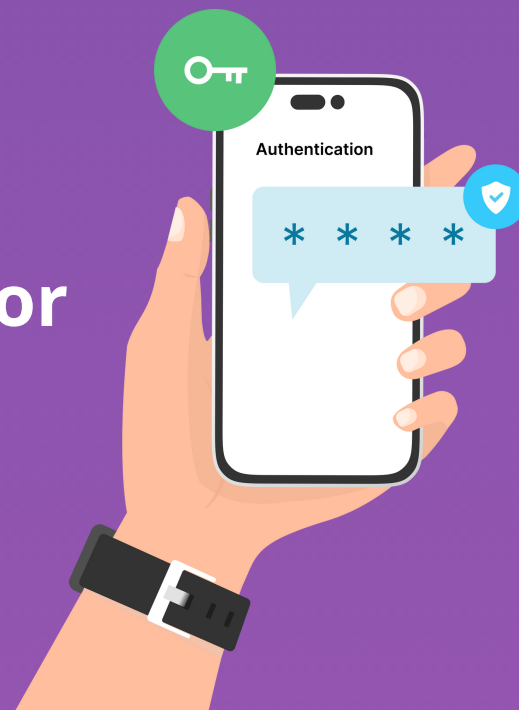
- One ROOT user is created by default
- Has **full administrative access** and privileges over all resources within the AWS account



Attackers with root credentials can do much more damage than when having restricted permissions

## Multi-Factor Authentication

- **Second authentication factor** in addition to user name and password





# IAM ROOT User



Root user should **not be used for everyday tasks**, even administrative ones

## Tasks only root user of an account can perform

- Activate IAM access to Billing and Cost Management
- Close account
- Change account settings
- ...



**Use Root only for such special cases**

# Administrative User

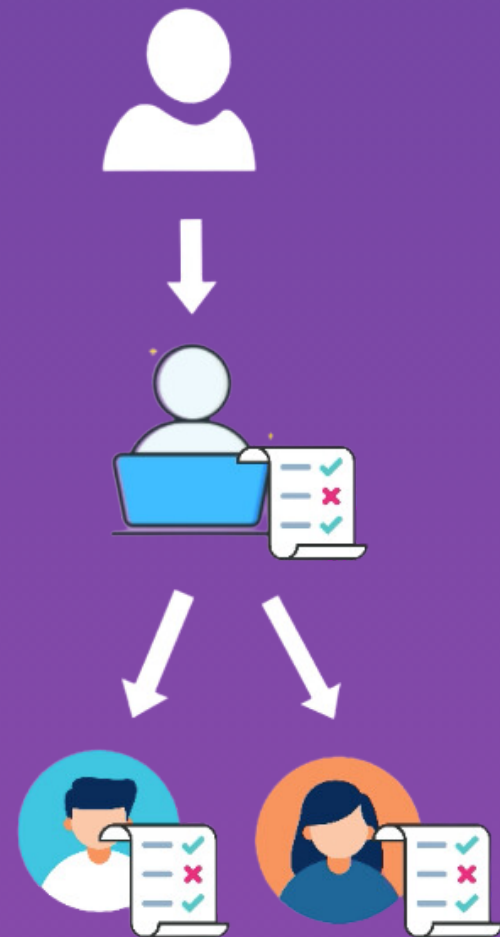
- To administer the AWS account, you should create an administrative user with less permissions
- Then **use the admin user to perform administrative tasks** including access management



1 -  
After AWS account sign up, you  
get root user

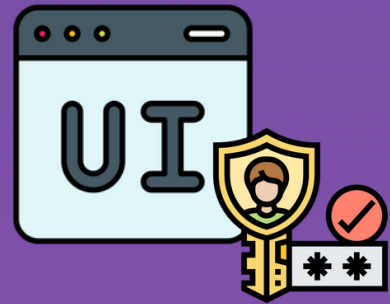
2 -  
Create user with  
administrative permissions

3 -  
Create other users with least  
privilege



# IAM Users

- There are 2 types of access:



## AWS Management Console

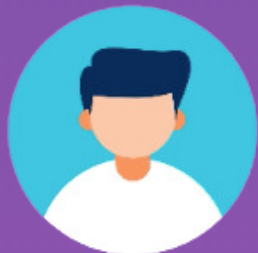
- Email and Password



## AWS Command Line Interface

- Access Key Pair

- Select which access to allow for the user



- System users, like GitLab CI will need programmatic access

# IAM Policies

- You manage access in AWS by **creating policies and attaching them to IAM identities (the Who)**

## What is an IAM Policy?

- Policy is a set of permissions that define **what actions** someone (who the policy is attached to) is able to perform on **which resources**
- You** can give permissions on a very **granular** basis
- Ability to limit further with conditions
- Policies are stored in AWS as JSON documents



## AWS Managed Policies

- Standalone policies that are created and administered by AWS
- Includes permissions for many common use cases

A white box containing JSON code for an IAM policy. The code is as follows:

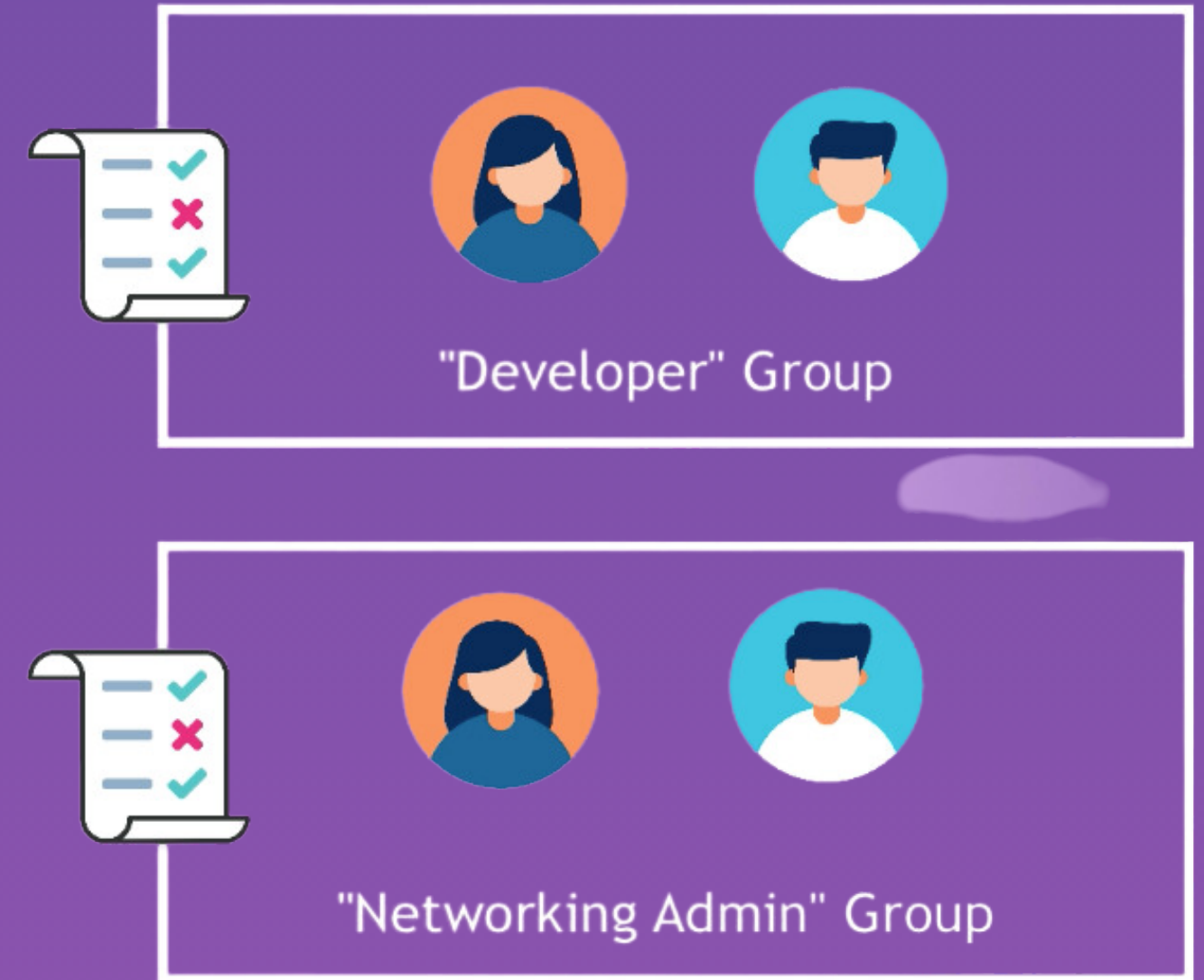
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DescribeInstances",
      "Resource": "*"
    }
  ]
}
```

Orange boxes highlight the following parts of the JSON: the "Effect" field of the first statement, the "Action" array of the first statement, the "Resource" field of the first statement, the "Condition" object of the first statement, the "Effect" field of the second statement, the "Action" field of the second statement, and the "Resource" field of the second statement.

# IAM Groups

## What is an IAM Group?

- You can organize users into groups
- You can **assign policies to the group** instead of individual users, making it easier to manage permissions for multiple users with similar access needs
- Any user, who is part of the group, will inherit the permission(s)

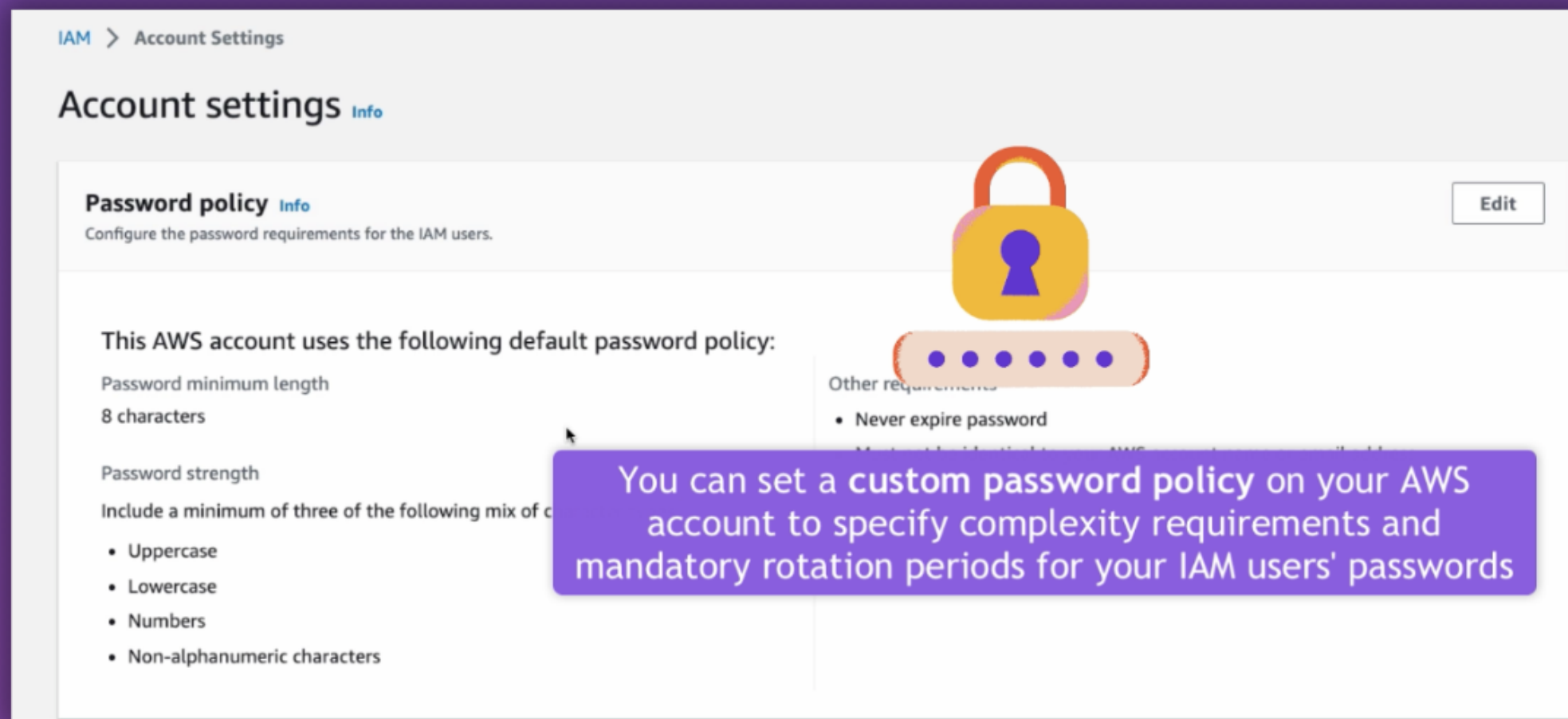




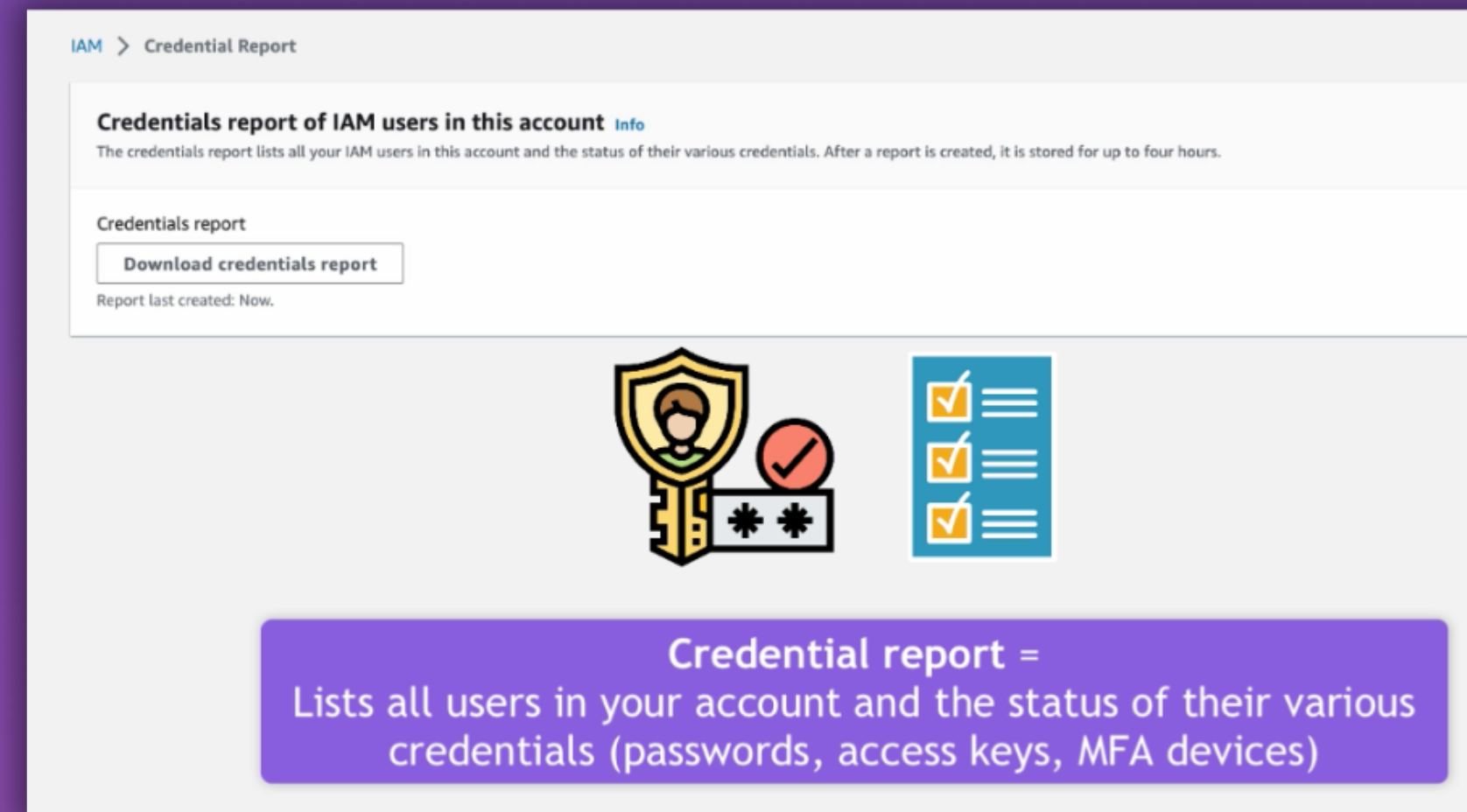
# Password Policy and Credentials Report

- There are 2 useful functionalities for AWS administrators

## Set up a Password Policy



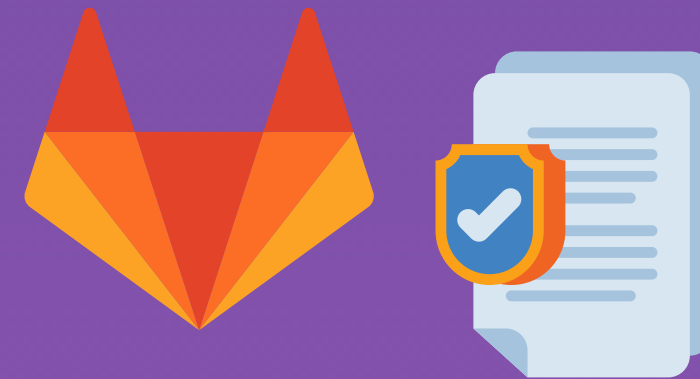
## Access Credentials Report



# Create GitLab User to perform CI/CD tasks

## Why create a dedicated GitLab User?

- Security Best Practice is to only give enough permissions needed by the service to perform its tasks. Not more.
- GitLab needs relevant access to be able to:
  - interact with AWS ECR repository
  - deploy to EC2 instance
  - **BUT nothing more!**
- For that, we want to create own dedicated user that has these restricted permissions



## Use GitLab Credentials in CI/CD

- Now instead of using Root access keys, we can replace it with the restricted GitLab access keys

# IAM Roles

## What is an IAM Role?

- Offers a secure way to grant permissions to entities, without the need for long-term access keys
- Roles are often used for services or instances that need to access other services within the cloud environment



## Key aspects of IAM roles:

### AWS Identity

- Similar to an IAM user, in that it is an **AWS identity with permission policies**

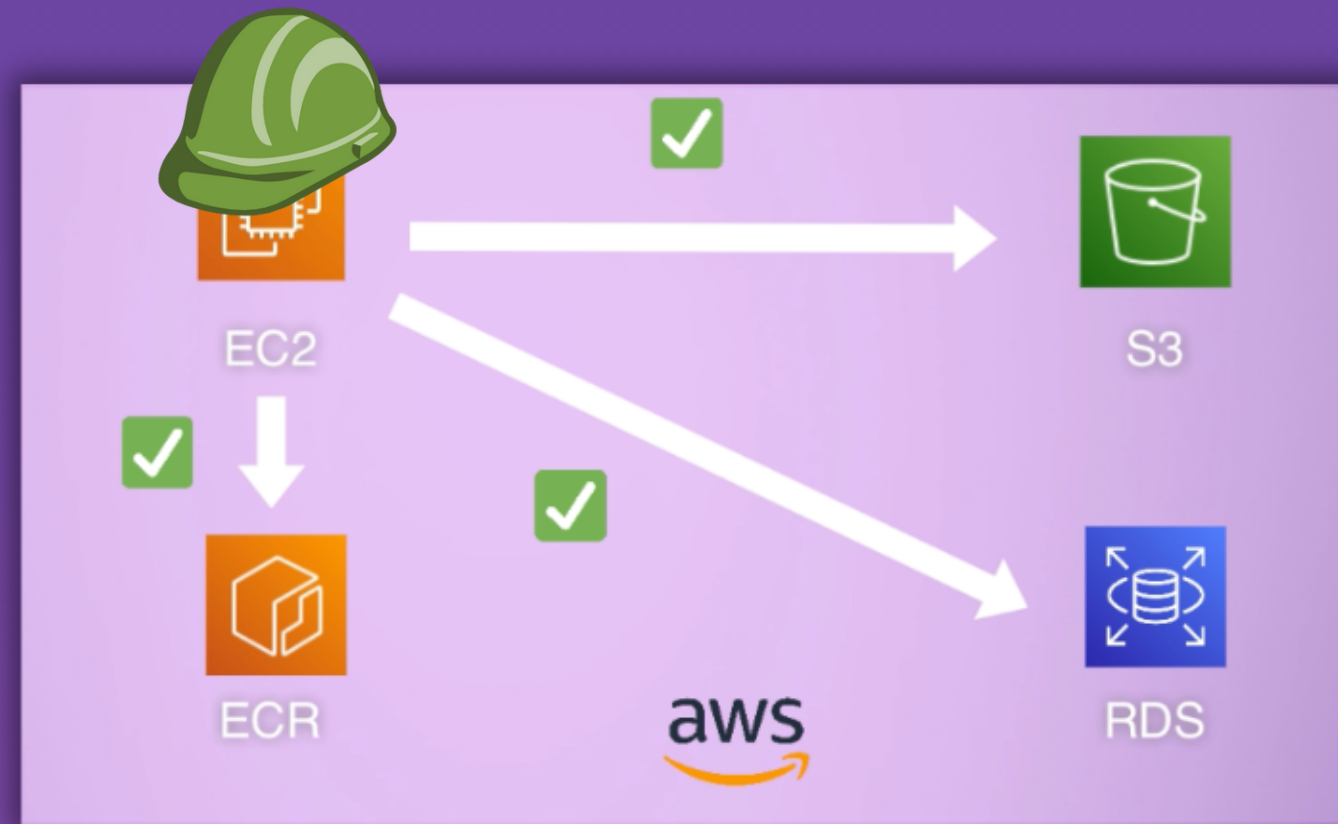
### Assumption

- **Entities assume roles** when they need to perform certain actions on resources

### No Permanent Credentials

- **Temporary credentials are dynamically generated** when the role is assumed

EC2 can assume role when needed



# IAM Roles

## Additional Security Layer

- Main use case is for AWS resources, but roles can also be used to delegate access to IAM users

- ✓ Easily revoke access
- ✓ Centralized access control
- ✓ Reduced credential exposure

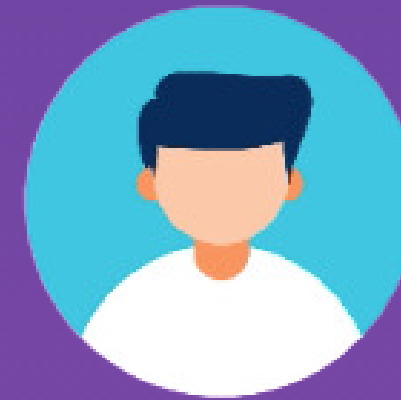


Assigning Role

# IAM Roles



Both are identities



- Temporary credentials
- Can be consumed temporarily by trusted entities
- There is a list of AWS managed roles

- Long-term credentials

## Trust Policy

- Roles have trust policy that defines which entities are allowed to assume the role



# IAM Roles

## Explain Roles via Hat analogy

- Wearing different hats depending on **your role** or what you are trying to achieve



- ✓ Temporarily assume role when needed