*“However beautiful the strategy, you should occasionally look at the results.”*

CLOUD DAY 2022

# Define a correct IAM strategy: treat your security baseline like a product


Paolo Latella

Cloud Practice Manager - Claranet Italia

AWS Hero / AWS Authorized Instructor / Partner Ambassador

@LatellaPaolo

# Design principles for security in the cloud

1. Implement a strong identity foundation
2. Enable traceability
3. Apply security at all layers
4. Automate security best practices
5. Protect data in transit and at rest
6. Keep people away from data
7. Prepare for security events

# Design principles for security in the cloud

1

Implement a strong identity foundation

4

Automate security best practices

Security Baseline as a Service

# Setting of game rules – Choose your governance model

## Centralized Model

- The MSO/MSP work as service broker
- More work streams with a DevOps approach
- Work streams with compliance requirements related to both projects and organization
- Builders work on project’s requirement with limited autonomy
- Safety before Agility

## Decentralized Model

- The MSO/MSP work as facilitator
- More work streams with a DevOps approach
- Work streams with compliance requirements more related to the organization.
- The project's compliance requirements are delegated to builders
- Balance Agility and Safety

Implement the **principle of least privilege** and enforce **separation of duties** with appropriate authorization for each interaction with your AWS resources. **Centralize identity management**, and aim to **eliminate reliance on long-term static credentials**.

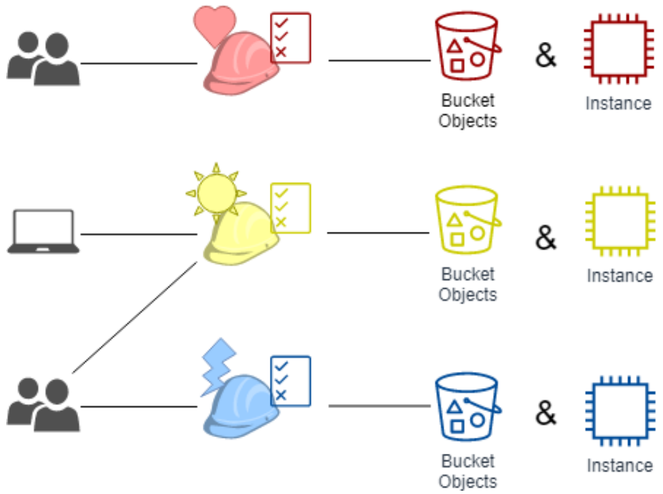
Photo by Alexandre Chambon on Unsplash

# Principle of least privileges

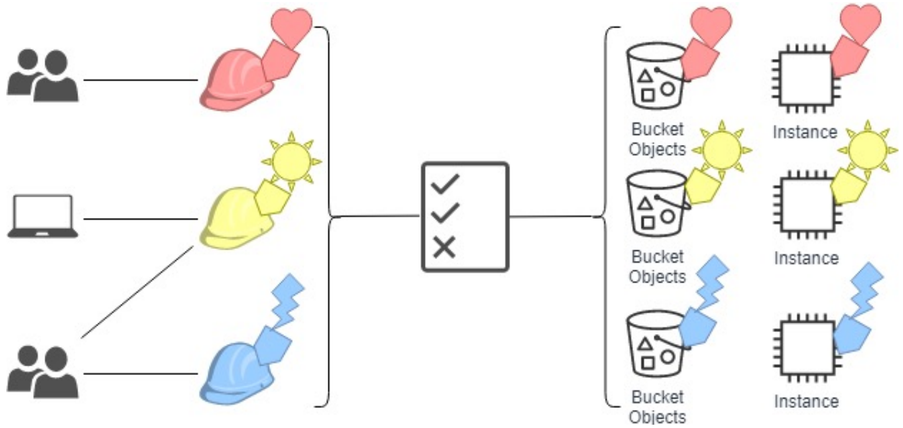
- Prefer ABAC to RBAC authorization model
- Adopt behaviour driven policy design
- Work with permission guardrails
- Introduce check-point at all level
- Separate workload using multi-account strategies

# Least Privileges - Prefer ABAC to RBAC model

**RBAC** defines permissions based only on a person's job function




**ABAC** defines permissions based on attributes of a person and the resource



# ABAC Model – Example 1

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "StartStopIfTags",
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:region:account-id:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "DataAnalytics",
                    "aws:PrincipalTag/Department": "Data"
                }
            }
        }
    ]
}
```

# ABAC Model – Example 2

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "ec2:startInstances",
            "ec2:stopInstances"
        ],
        "Resource": "*",
        "Condition": {"StringEquals":
            {"aws:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"}}
    }
}
```

# Least Privileges - Adopt behaviour driven policy design

- Your developers are working on a new application
- Your application is in development environment
- To encourage experimentation you've configured IAM policy with broad permissions.
- Now the application is ready for production

- Your developers are working on a new application
- Your application is in development environment
- To guarantee the right level of security you've configured IAM policy with less or no permissions
- Analyze API activity log
- Create policy based on activity
- Iterate
- Now the application is ready for production

# Least Privileges - IAM Access Analyzer

IAM Access Analyzer continuously monitors and analyzes resource policy to help you understand the potential security implications. It guides you toward least privilege permissions.

- Identify resources in your organization and accounts that are shared with an external entity.
  - Least Privileges focus on the borders
- Validates IAM policies against policy grammar and best practices.
  - Least Privileges focus on policy grammar
- Generate IAM policies based on access activity in your AWS CloudTrail logs.
  - Least Privileges focus on behaviour

```
maverick@Paolos-MacBook-Pro-2 CCMDLabs % aws accessanalyzer validate-policy --policy-type RESOURCE_POLICY --profile CCMD-EUCOM
t-PaoloLatella --policy-document file://./ValidatePolicy.json --region us-east-1
{
    "findings": [
        {
            "findingDetails": "Using Allow with NotPrincipal can be overly permissive. We recommend that you use Principal ins
            "findingType": "SECURITY_WARNING",
            "issueCode": "ALLOW_WITH_NOT_PRINCIPAL",
            "learnMoreLink": "https://docs.aws.amazon.com/IAM/latest/UserGuide/access-analyzer-reference-policy-checks.html#ac
zer-reference-policy-checks-security-warning-allow-with-not-principal",
            "locations": [
                {
                    "path": [
                        {
                            "value": "Statement"
                        },
                        {
                            "index": 0
                        },
                        {
                            "value": "Effect"
                        }
                    ],
                    "span": {
                        "end": {
                            "column": 25,
                            "line": 4,
                            "offset": 76
                        },
```

# IAM Access Analyzer - Generate

```
aws accessanalyzer start-policy-generation \
--policy-generation-details principalArn=arn:aws:iam::278014156012:user/paolo.latella \
--cloud-trail-details '{"trails":[…],"startTime":"2022-10-15T10:30:00.000Z","endTime":"2022-10-15T18:30:00.000Z"}'
--profile CCMD-EUCOM-ThirdFleet-PaoloLatella \
--region eu-west-1
```

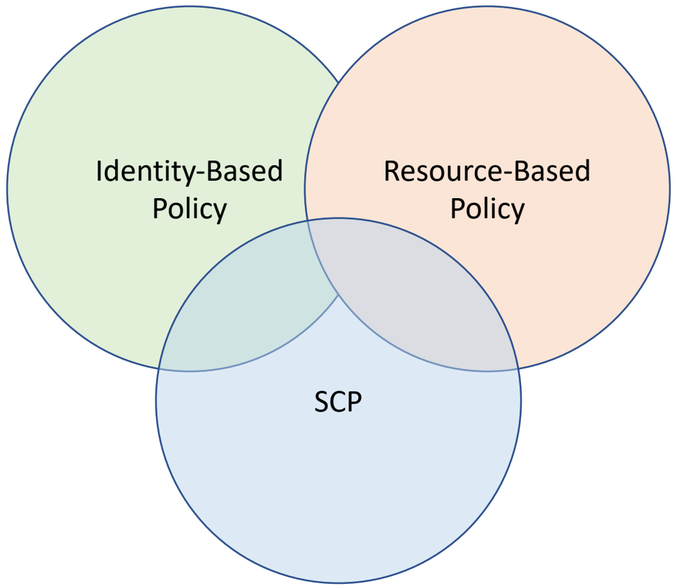
```
aws accessanalyzer get-generated-policy \
--job-id jobid \
--profile CCMD-EUCOM-ThirdFleet-PaoloLatella \
--region eu-west-1
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/shared/*"
            ]
        }
    ]
}
```

# Least Privileges – Work with permissions guardrails

*In any case, we must define a set of controls that prevents builders from going off-road (guardrails)*

# Work with permissions guardrails – SCP




Deny List Strategy - Actions are allowed by default, and you specify what services and actions are prohibited

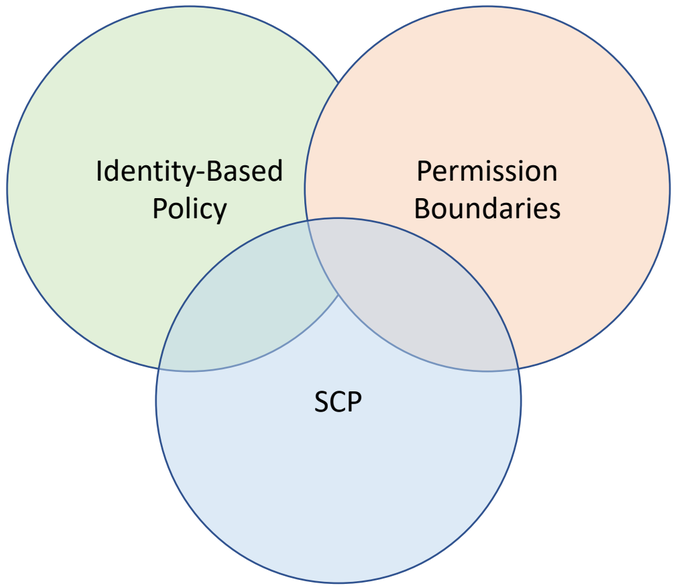
- **Require less maintenance**
- Policy require less space (max 5KB)

Allow List Stratey - Actions are prohibited by default, and you specify what services and actions are allowed

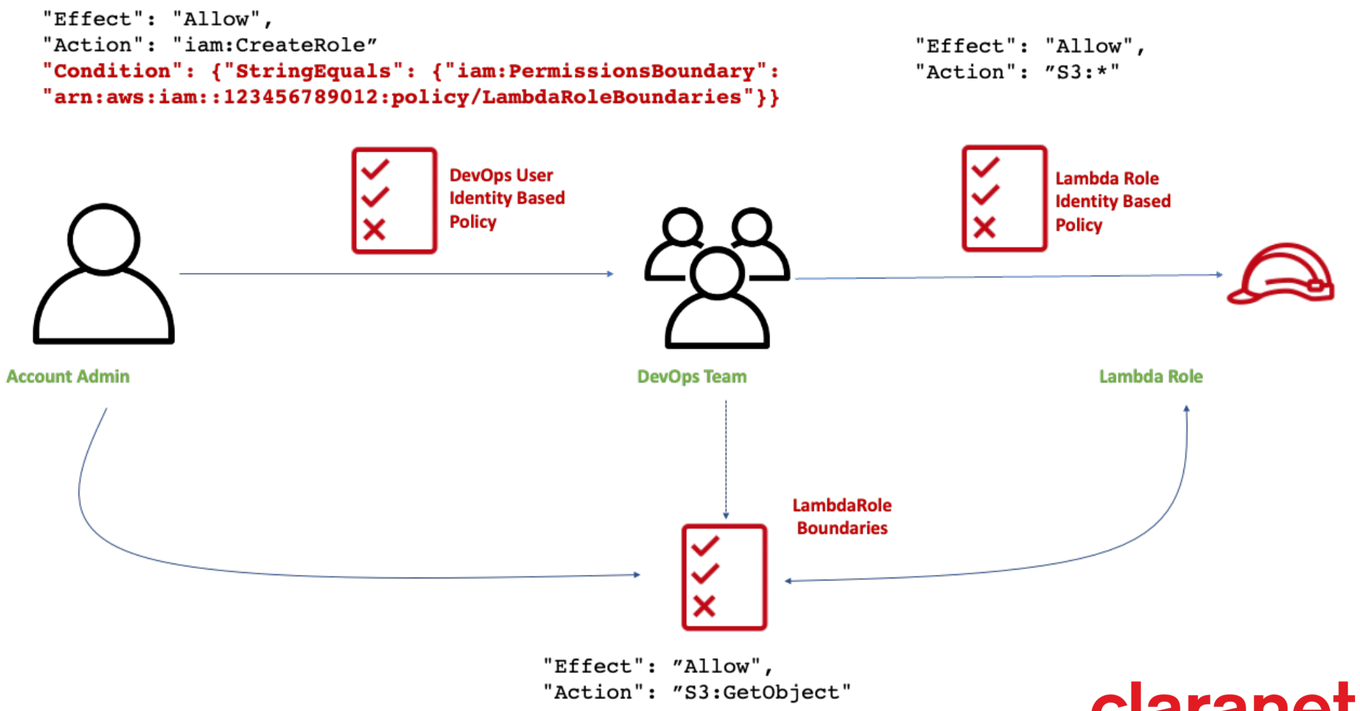
- The implicit deny impact on existing resources
- **Most aligned to principle of least privileges**

**Deny List + Exception**
- NotAction
- NotResource
- NotPrincipal
- StringNotLike
- ArnNotLike

# Permission Guardrails – Preventive Control - Permission Boundaries




Delegate + Least Privileges = Prevent privilege escalation




claranet

# Separation of Duties

Separation of duties or Segregation of duties (SOD) refers to the principle that no user should be given enough privileges to misuse the system on their own. For example, the person authorizing a paycheck should not also be the one who can prepare them – NIST Glossary.

- Separation of duties can be enforced either statically (by defining conflicting roles) or dynamically (by enforcing the control at access time).
- Separation of duties can be approached as sequential separation (two signatures principle), individual separation (four eyes principle), spatial separation (separate action in separate locations), factorial separation (several factors contribute to completion)

# Separation of duties

## Static SOD

- Best fit on RBAC model
- Secure Control Policies (SCP)
- Permission Boundaries

## Dynamic SOD

- Best fit on ABAC model
- Temporary permission by STS
- Session Policies

## Sequential SOD

- Codepipeline with approval actions
- Codecommit and Pull request
- Cloudformation WaitCondition/WaitConditionHandle

## Individual SOD

- Role Trust Relationship
- VPC Peering / Transit GW Attachment
- Codepipeline with more approval actions
- Codecommit with more approvals needed

## Spatial SOD

- Multi Account Strategies with Assume Role

## Factorial SOD

- Require MFA to perform action

# Dynamic SOD – Assume role

```
aws sts assume-role --role-arn "arn:aws:iam::278014156012:role/DevOpsPutInProduction" --role-session-name SOD-Demo-Session --profile DevSecOps --source-identity <username>
```

```
aws wafv2 associate-web-acl --web-acl-arn arn:aws:wafv2:eu-west-1:278014156012:regional/webacl/TestDynamicSOD/45289a92 --resource-arn arn:aws:elasticloadbalancing:eu-west-1:278014156012:loadbalancer/app/TestDynamicSOD/76da635f1d7d864d
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SetAwsUserNameAsSourceIdentity",
            "Effect": "Allow",
            "Action": "sts:SetSourceIdentity",
            "Resource": "arn:aws:iam::278014156012:role/CCMD-DevOpsPutInProduction",
            "Condition": {
                "StringLike": {
                    "sts:SourceIdentity": "${aws:username}"
                }
            }
        },
        {
            "Sid": "PermitWriteOnWebACL",
            "Action": [
                "wafv2:CreateWebACL",
                "wafv2:DeleteWebACL",
                "wafv2:UpdateWebACL"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

DevSecOps User (Policy)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1665562473928",
            "Action": [
                "wafv2:AssociateWebACL",
                "elasticloadbalancing:SetWebACL"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:wafv2:eu-west-1:278014156012:regional/webacl/*",
                "arn:aws:elasticloadbalancing:eu-west-1:278014156012:loadbalancer/app/*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:ResourceTag/Owner": "${aws:SourceIdentity}"
                },
                "StringEquals": {
                    "aws:ResourceTag/Environment": "Production"
                }
            }
        }
    ]
}
```

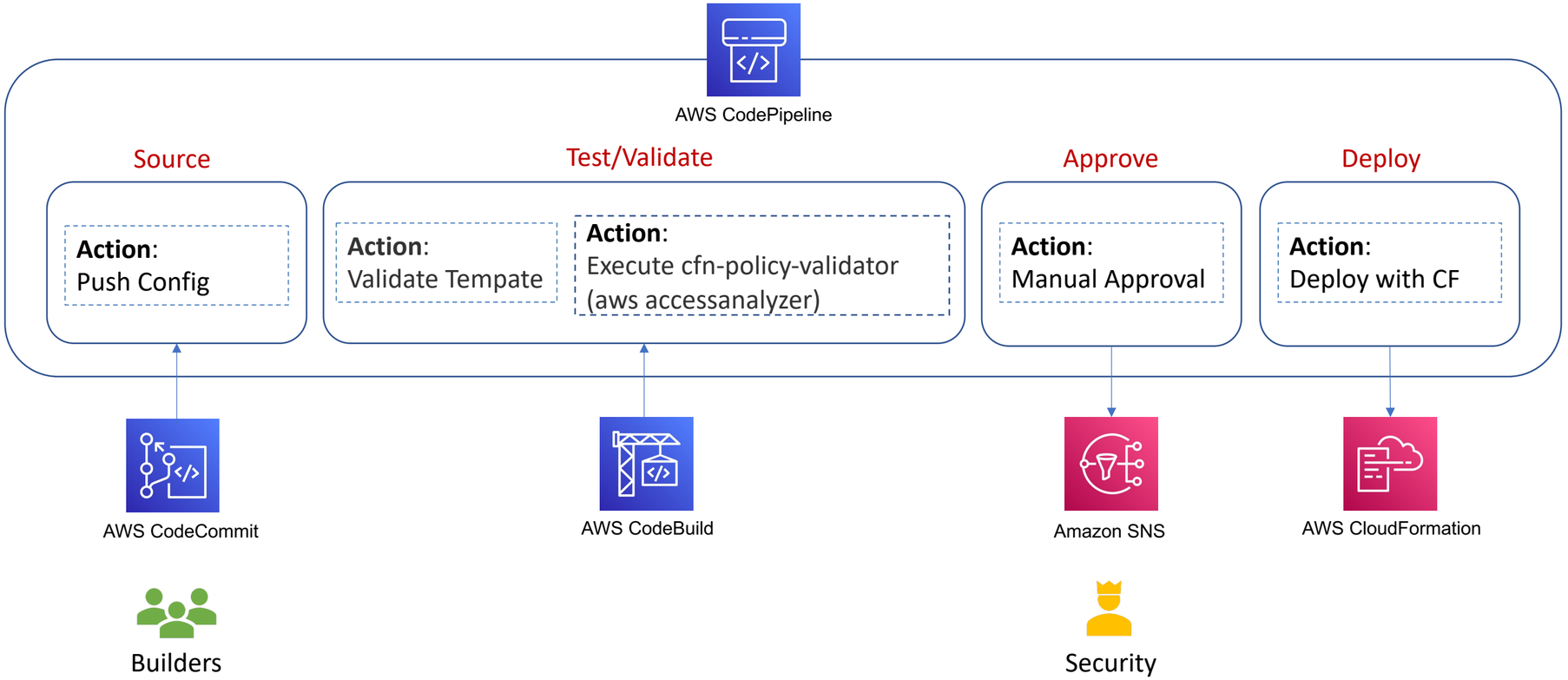
DevOpsPutInProduction Role (Policy)

Don’t forget SCP to:

- Require TAG with keys **Environment** and **Owner** on resource creation
- Require **SourceIdentity**=Username in assume-role API

claranet

# Sequential SOD - CodePipeline



# Separation of duties – Start from risk

- There is no a “Silver bullet” that organizations can use to ensure separation of duties.
- A separation of duties plan starting from a risk-assessment workshop which involves the following steps:
    1. Identify the assets
    2. Decompose the process that involves the asset
    3. Put in evidence the action-entity relationship
    4. Identify Collision that could produce a fraud or error
    5. Mitigate introducing one of the SOD practice

# SOD in DevOps/DevSecOps

- Is the SOD practicable on a DevSecOps approach ?

# SOD in DevOps/DevSecOps

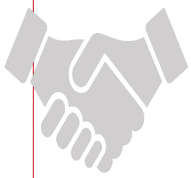
- Is the SOD practicable on a DevSecOps approach ?
- DevSecOps posits that all participants in the development cycle have shared responsibility for the security of the application and its environment.
  - Shared responsibility -> different duties -> same team
- Even in cases where roles are initially played by the same people, it’s important to start with a separate set of functional roles (leveraging AWS Roles)
  - Permit to work with some kind of SOD
  - Simplify the adoption of SOD in the future


Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. Create secure architectures, including the **implementation of controls that are defined and managed as code** in version-controlled templates.

# Implementation of controls

- To meet the compliance program, the organization implements controls, defines procedures and writes guidelines that satisfy these policies.
- Use Compliance as Code to maintain security without sacrificing business agility.
  - Adopt a new approach to planning, developing, and testing policies
  - Define your compliance policies such that they can be treated as software.

The organization share responsibility with cloud service providers

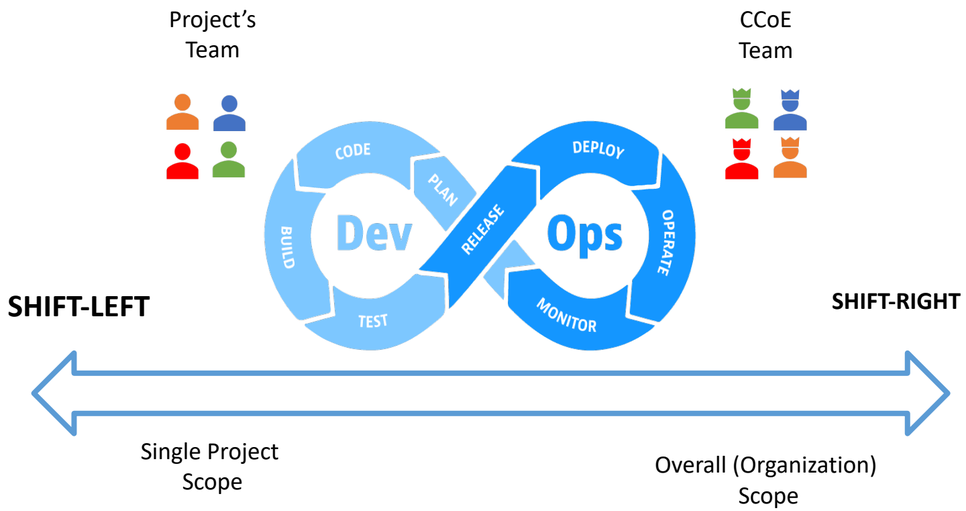


The organization must improve its business agility without compromising safety

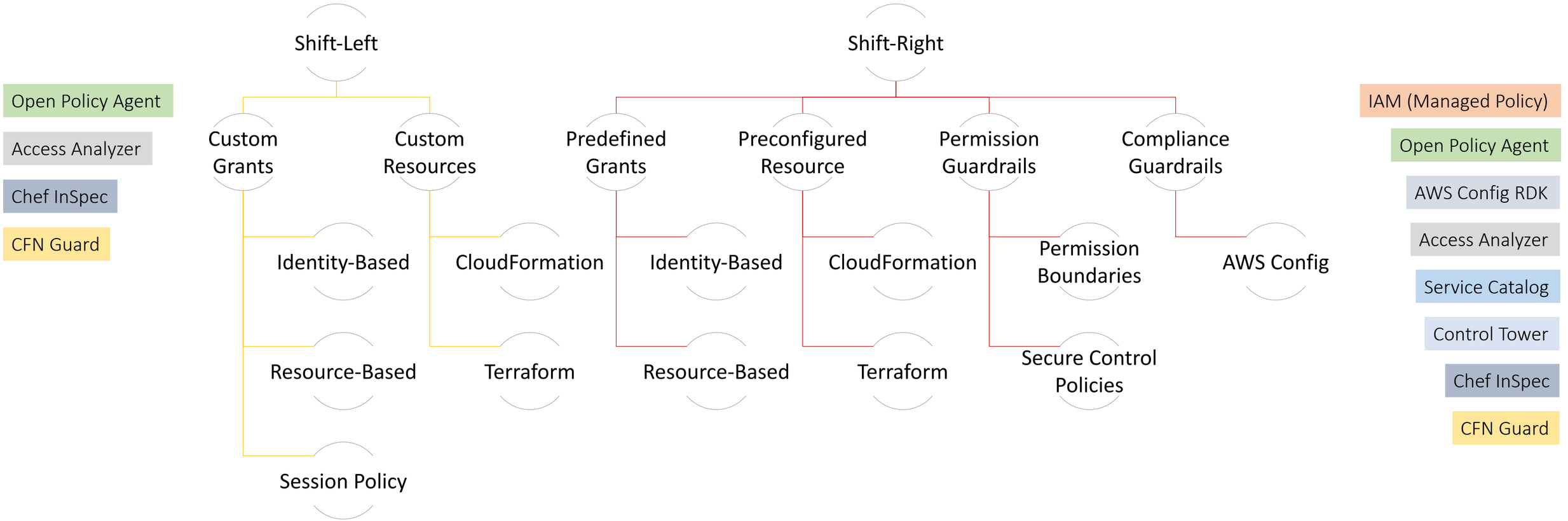
# Compliance as Code – Shift-Left vs Shift-Right

Encourage both shift-left and shift-right approach especially in a decentralized model

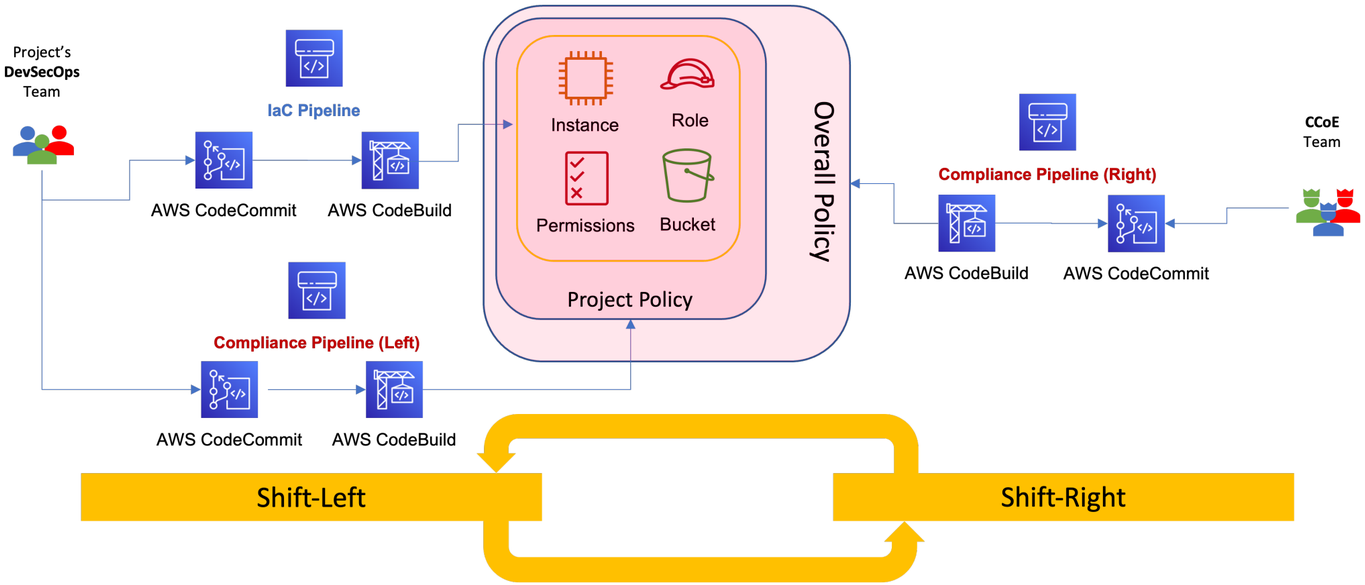
- Shift-Left
  - Detect compliance issues very early in the process
  - Our tests are more related to a specific workload.
  - Tests planned and executed by the project's team
  - More open-source and CSP agnostic tools
- Shift-Right
  - Require more effort to align with requirements
  - Requirements defined at the organization level.
  - The tests are planned and executed by a CCoE
  - Tools (services) provided directly by CSP

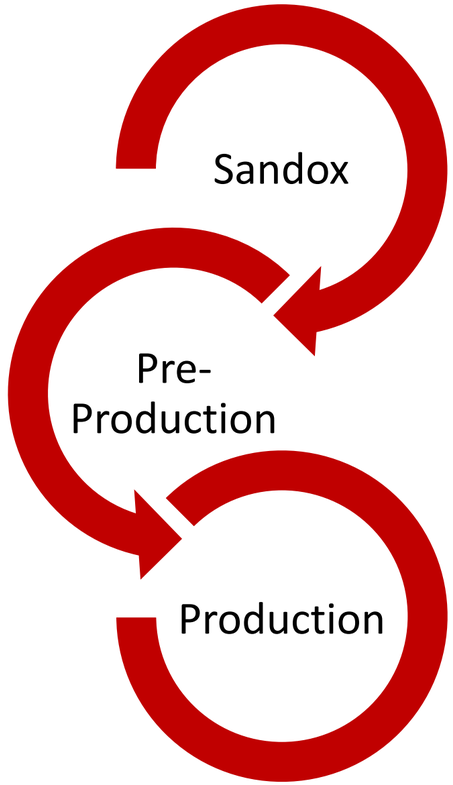


# Shift-Left vs Shift-Right – The right tools for the right job



# Shift-Left vs Shift-Right - Continuous compliance



# Shift-Left / Shift-Right – CFN Guard and OPA

- AWS CloudFormation Guard is a open source policy-as-code evaluation tool
  - Guard validate JSON/YAML data against Guard DSL policy rules
  - Guard doesn't validate templates for valid syntax or allowed property values (use cfn-lint)
- The Open Policy Agent (OPA) is an open source, general-purpose policy engine that unifies policy enforcement across the stack.
  - A graduated project in the Cloud Native Computing Foundation (CNCF) landscape
  - OPA Policies written in Rego
  - Daemon, Library, CLI Tool

```
maverick@Paolos-MacBook-Pro-2 CFNGuard % cfn-guard validate --rules rules.guard --data template.json
Other
template.json Status = FAIL
FAILED rules
rules.guard/verify_max_session_duration    FAIL
---
Evaluating data template.json against rules rules.guard
Number of non-compliant resources 1
Resource = MyTestRole {
  Type      = AWS::IAM::Role
  Rule = verify_max_session_duration {
    ALL {
      Check =  %roles[*].Properties.MaxSessionDuration LESS THAN EQUALS  3600 {
        ComparisonError {
          Error           = Check was not compliant as property value [Path=/Re
] Value=4600] less than equal to value [Path=[L:0,C:0] Value=3600].
          PropertyPath    = /Resources/MyTestRole/Properties/MaxSessionDuration[L:9,C:35]
          Operator        = LESS THAN EQUAL
          Value           = 4600
          ComparedWith    = 3600
          Code:
                7.            "AssumeRolePolicyDocument" : "RomePolicy",
                8.            "Description" : "Test Cloudformation Guard",
                9.            "MaxSessionDuration" : 4600,
               10.            "PermissionsBoundary" : "arn:aws:iam::account:policy/PermissionsBoundary",
               11.            "RoleName" : "TestRole"
               12.          }
    }
  }
}
maverick@Paolos-MacBook-Pro-2 CFNGuard %
```

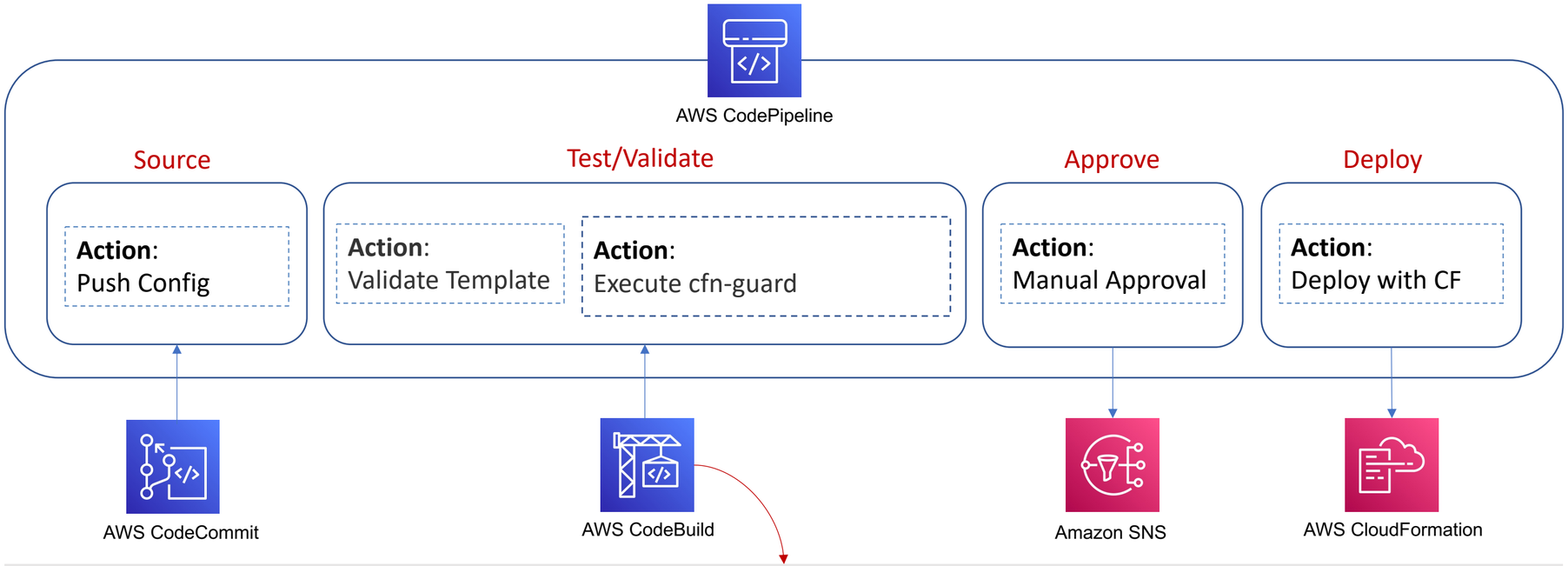
```
let roles = Resources.*[Type == "AWS::IAM::Role"]
let ec2 = Resources.*[Type == "AWS::EC2::Instance"]

rule verify_max_session_duration when %roles !empty {
    %roles.Properties.MaxSessionDuration <= 3600
}

rule verify_tags_presence when %ec2 !empty {
  %ec2.Properties.Tags !empty
}
```

cfn-guard validate --rules rules.guard --data template.json

# Cloudformation Guard in CI/CD Pipeline




```
version: 0.2
phases:
    install:
        commands:
        - curl https://raw.githubusercontent.com/aws-cloudformation/cloudformation-guard/main/install-guard.sh | sh
    build:
        commands:
         - cfn-lint template.json
         - cfn-guard validate --rules rules --data template.json
```

# Takeaways

- Adoption of a decentralized model permits to strike the right balance between agility and safety, but requires the design and implementantion of a security baseline
- Implement a strong identity foundation adopting the principle of least privilege and enforce separation of duties with the appropriate authorization for each interaction with your AWS resources.
- Automate security best practices in creating secure architectures, including the implementation of controls that are defined and managed as code in version-controlled templates.

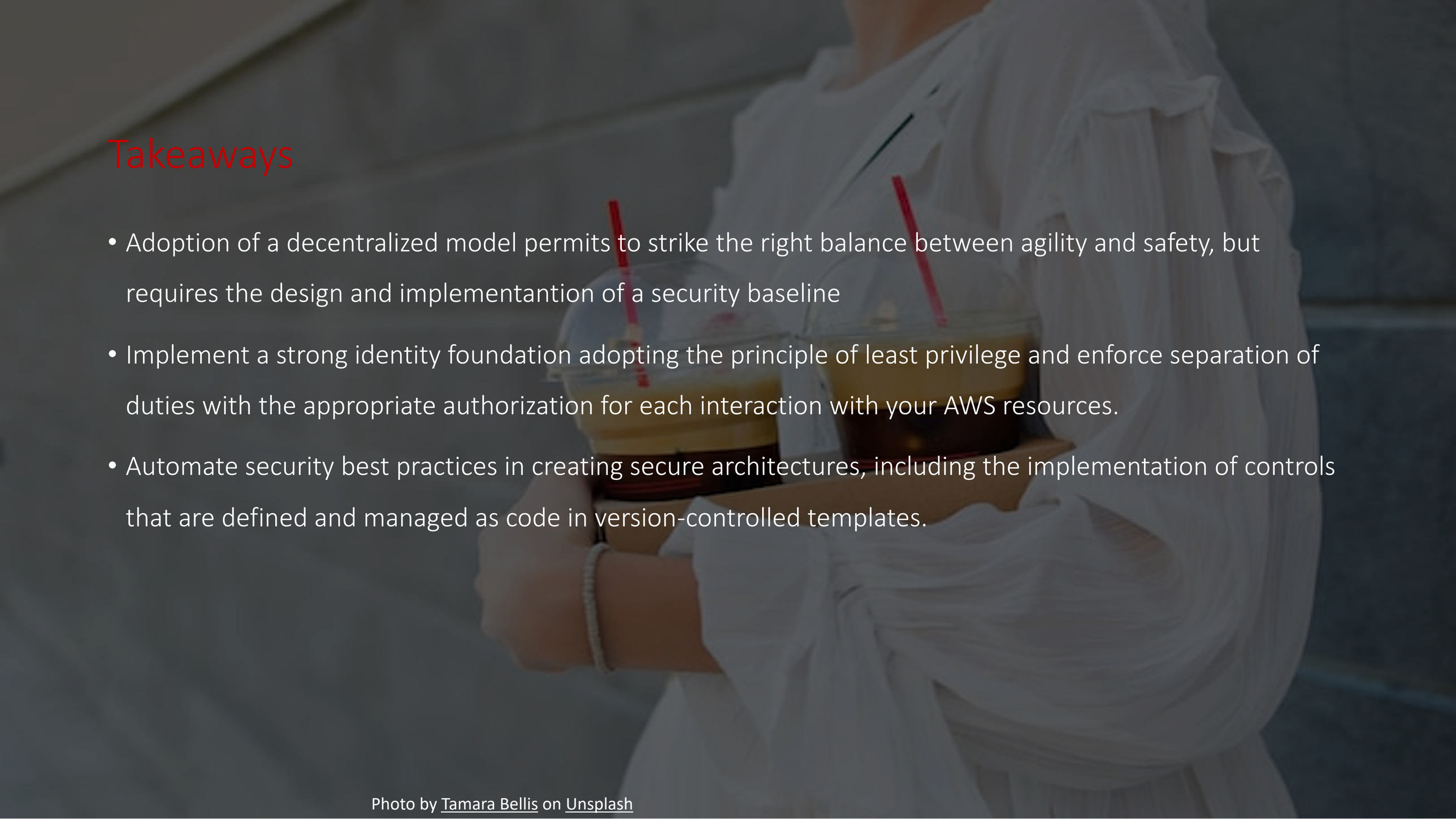


Photo by Tamara Bellis on Unsplash

# SPONSOR

## DIAMOND

beSharp. we make IT run.

GUCCI

splunk>

TD SYNNEX

## PLATINUM

improove

.NET on aws

## GOLD

CODICEPLASTICO

Cubbit

PARTNER

AWS
User Groups
Milano


Microsoft
Azure

UGIdotNET

# Grazie!