algosec

Secure application connectivity.
Anywhere.

AWS best practices:
Essential security for organizations managing multiple AWS accounts

An AlgoSec Whitepaper

Cloud adoption offers significant benefits but exposes organizations to increased security risks, including data breaches, compliance failures, and misconfigurations. These risks are amplified in multi-account AWS environments, which, while offering advantages like resource isolation and granular access control, introduce complexities in managing security policies, cross-account access, and monitoring.

This white paper outlines essential security best practices for organizations managing multiple AWS accounts. A multi-account setup allows for separate environments for different teams or projects, enabling centralized control and governance. AWS Organizations facilitate this by simplifying billing and access management while allowing for the grouping of accounts into Organizational Units (OUs) with distinct policies. This structure, combined with tools like service control policies (SCPs) and identity and access management (IAM) roles, enhances security and compliance in complex cloud environments.

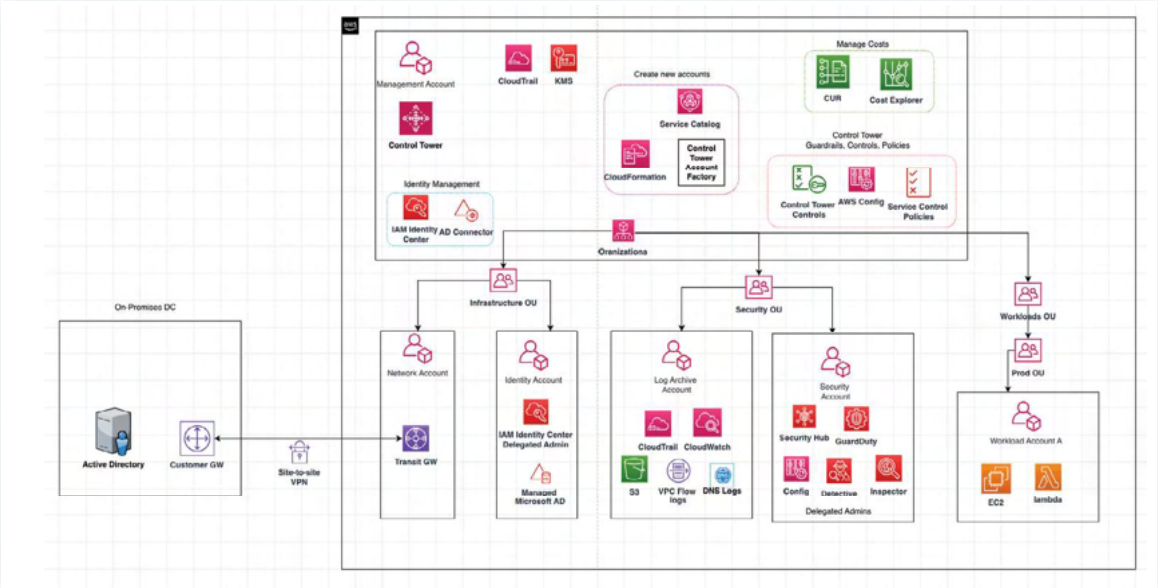


Figure 1: AWS multi-account setup (Source: Spacelift)

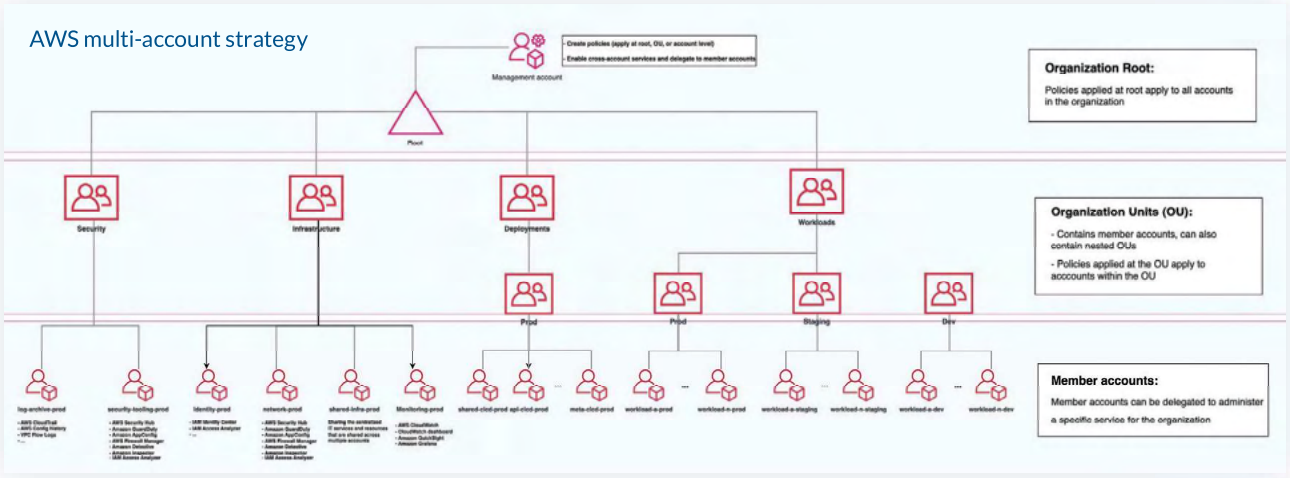


Figure 2: AWS multi-account strategy, showing how different teams and environments are organized using OUs (Source: Towardsaws.com)

Benefits of a multi-account setup

There are various benefits of a multi-account setup:

- Workload isolation for dev/test/prod, reducing accidental changes and protecting sensitive data
- Granular access control, so each account can have unique IAM policies that guarantee specific user access and uphold the least privilege principle
- Streamlined billing for improved transparency into spending and project tracking by grouping expenses under a single AWS organization
- Reduced attack surface, limiting risk to other accounts and enhancing incident response for an enhanced security posture

Security challenges in multi-account architectures

- While a multi-account architecture offers significant advantages, it also introduces unique security challenges that organizations must address.
- Complex policy management across accounts can lead to inconsistencies and security gaps.
- Cross-account access can create vulnerabilities if IAM roles or permissions are misconfigured risking exposure of sensitive resources.
- Centralized monitoring is an absolute must to keep track of security events across several accounts and identify questionable activity.
- Inadequate cross-account permissions controls can result in inadvertent data leaks particularly when teams are not aware of them.

Best practices for securing AWS multi-account architectures

To mitigate the above challenges, organizations should adopt the following best practices.

Centralized governance: Utilizing AWS Organizations enables effective management of multiple accounts. Create OUs to group accounts according to function, team, or project. Also, implement SCPs to enforce governance and compliance requirements consistently across accounts.

Strong identity and access management: Implement IAM roles for cross-account access rather than using static IAM users. This minimizes risks associated with credential management and enables organizations to enforce fine-grained permissions tailored to the needs of each account. Additionally, making sure multi-factor authentication (MFA) is required for all accounts increases security beyond a mere password.

Continuous monitoring and auditing: Use AWS CloudTrail to log all API calls across accounts. This provides a comprehensive audit trail for forensic analysis and compliance verification. Implement AWS Config to continuously monitor resource configurations and compliance, allowing for immediate identification of any deviations from security policies.

Network security best practices: Design isolated Virtual Private Clouds (VPCs) for different accounts to reduce the risk of lateral movement within your environment. Use security groups and network access control lists (NACLs) to define strict inbound and outbound traffic rules, controlling access to resources effectively.

Automated incident response: Leverage AWS Lambda to enable automated responses to security incidents. For instance, you can create Lambda functions that trigger specific events, such as isolating compromised resources or alerting administrators about suspicious activity. Integration with AWS Security Hub will provide a centralized view of security alerts across accounts, facilitating a quicker response to potential incidents.

Regular security reviews and updates: Conduct regular audits of security policies and configurations across all accounts. Establish a cadence for reviewing IAM roles and SCPs to ensure they remain aligned with evolving organizational needs and industry best practices.

Managing accounts

With multiple AWS accounts, effective management is key to ensuring security and efficiency. AWS Organizations provides a centralized platform for managing accounts, streamlining billing, simplifying access management, and enforcing governance policies.

Key features of AWS organizations include:

- **Consolidated billing:** Simplifies financial oversight and tracking
- **Centralized access management:** Streamlines account provisioning and policy application
- **Organizational units (OUs):** Enable grouping of accounts for applying specific policies and controls
- **Service control policies (SCPs):** Define permission boundaries across accounts to enforce governance and compliance

Using SCPs effectively

SCPs allow organizations to define which actions are permitted or denied across accounts. For example, an SCP can prevent users from deleting S3 buckets or restrict access to sensitive resources.

Best practices for SCPs:

- Test policies in a staging environment before implementation.
- Maintain comprehensive documentation of all SCPs.
- Conduct regular reviews to ensure SCPs meet evolving needs.

By leveraging AWS Organizations and SCPs, organizations can establish a strong foundation for secure and efficient management of their multi-account AWS environment.

Controlling access

Controlling access to resources in AWS is key to achieving a secure environment. Organizations thus must leverage the right mechanisms to make sure only authorized parties can interact with specific AWS services and data. This section delves into the principles and best practices for controlling access in AWS, focusing primarily on AWS Identity and Access Management and IAM roles.

Identity and access management

AWS IAM controls resource access and user identities while upholding security and legal requirements. It reduces unwanted access by limiting permissions to solely those resources required for a particular role or task (the principle of least privilege).

Best practices for IAM:

Implement least privilege: Regularly review and refine user permissions to ensure access has only been granted to resources required for a given role. For example, developers whose work only requires access to specific S3 buckets should have their permissions limited accordingly. This practice is crucial for minimizing the attack surface and adhering to regulatory standards, as outlined in NIST SP 800-53.

Enable MFA: MFA mandates users provide extra verification, like a code sent via SMS, alongside their password. This is particularly critical for accounts with administrative access. The AWS IAM MFA documentation can help guide you in implementing MFA effectively.

Use IAM groups: Organizing users into IAM groups simplifies permission management. Granting permissions to groups—not individual users—ensures consistency and ease of management. AWS' documentation on IAM best practices provides additional information.

Perform regular audits: Conducting regular audits of IAM roles, users, and policies helps organizations identify and revoke unnecessary permissions. Regular audits are a key component of maintaining a secure IAM environment and aligning with AWS security best practices.

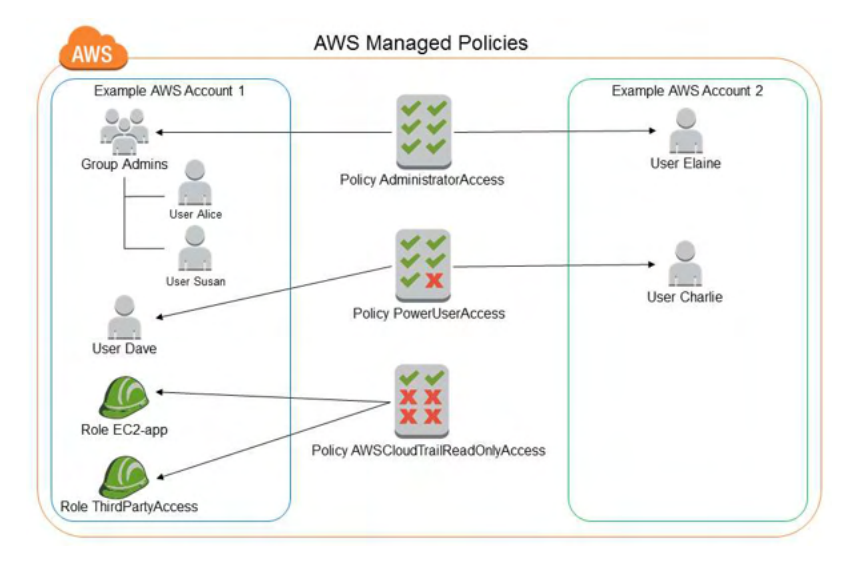


Figure 3: AWS managed policies (Source: AWS documentation)

IAM roles

IAM roles grant permissions to entities to perform specific actions on AWS resources, enabling secure cross-account access and service delegation.

Key features include temporary credentials, which provide limited-time access to reinforce security, and cross-account access, which allows users or services from one AWS account to interact with resources in another without duplicating user accounts; this is particularly useful for organizations managing multiple AWS accounts.

Example of creating IAM roles

When creating an IAM role, specific permissions and a trust policy need to be defined. Below is an example an IAM role policy that allows EC2 instances to assume it:

```
resource "aws_iam_role" "example" {
  name = "example-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
    {
    Action = "sts:AssumeRole",
    Effect = "Allow",
    Principal = {
          Service = "ec2.amazonaws.com"
    }
    }
    ]
  })
}
```

In this example, EC2 instances are allowed to assume an IAM role, meaning they can perform actions defined in the role's policy. This role-based approach is essential for minimizing security risks associated with hardcoding AWS credentials in applications.

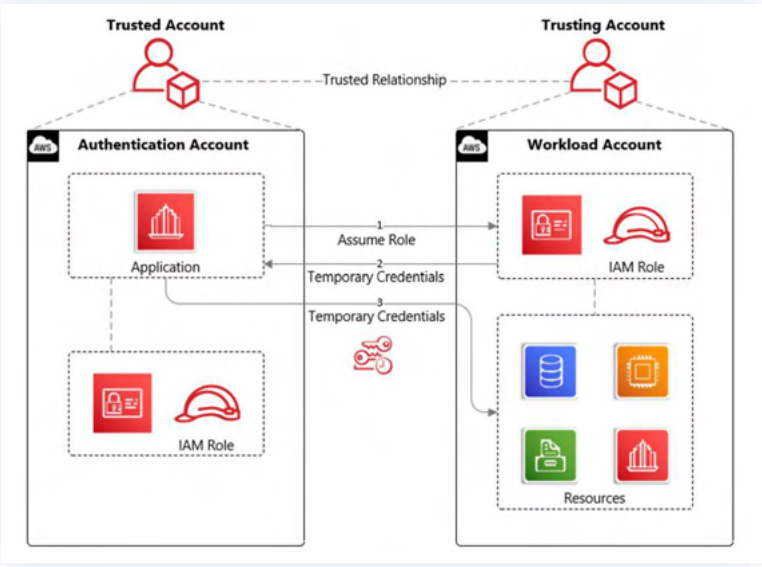


Figure 4: How IAM roles enable secure access across various AWS accounts and services (Source: Medium blog)

Monitoring and logging

Continuous monitoring and logging are essential for maintaining security in a multi-account AWS environment.

Key services for this include:

AWS CloudTrail: Provides an audit trail for security analysis and compliance by logging all API calls made in your AWS account.

AWS Config: Offers a detailed inventory of your AWS resources and their configurations, enabling continuous compliance monitoring.

By utilizing CloudTrail and Config, organizations can gain insights into user activity, track resource changes, and ensure adherence to security policies.

This section explores the functionalities of these services and how to best implement them.

AWS CloudTrail

AWS CloudTrail creates a record of any AWS API calls made in your account. It provides visibility into user activity, changes to resources, and actions taken by AWS services, thereby enabling organizations to audit their AWS usage and enhance security.

Key features of AWS CloudTrail include:

Comprehensive logging: CloudTrail logs include details about the API calls made, including IP address, call time, caller ID, and more. This information is essential for forensic analysis if a security incident occurs.

Multi-account support: CloudTrail can be configured to log API activity across multiple AWS accounts, providing a centralized view of all activities within an organization.

Example configuration of CloudTrail

Consider a scenario where an organization wants to set up CloudTrail to monitor all API calls in their environment. They create a CloudTrail trail that logs all API activity and stores the logs for later analysis:

```
resource "aws_cloudtrail" "example" {
  name                          = "example-trail"
  s3_bucket_name                = aws_s3_bucket.example.bucket
  include_global_service_events = true
}
```

In the above example, CloudTrail is configured to log events and store them in a specified S3 bucket. This setup ensures that all API activity is captured for auditing and compliance purposes.

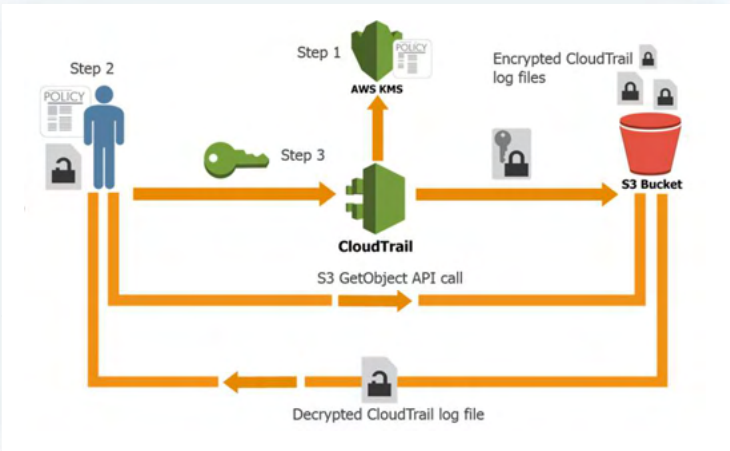


Figure 5: Components of AWS CloudTrail, showing how API calls are logged and stored for analysis (Source: LinkedIn Pulse)

AWS Config

AWS Config provides detailed visibility into the configurations of AWS resources in your account. It enables companies to determine their compliance with both internal policies and regulatory standards, monitor changes to resource configurations, and automate compliance checks.

Key features of AWS Config include:

- **Resource inventory:** AWS Config maintains an up-to-date inventory of your AWS resources, providing a historical view of resource configurations.
- **Compliance monitoring:** AWS Config allows you to create rules to evaluate the configurations of your AWS resources against best practices and compliance requirements.

Example configuration of AWS Config

In an organization where compliance with data protection policies is crucial, AWS Config can be set up to ensure that S3 bucket versioning is enabled. This is achieved by creating a compliance rule within AWS Config:

```
resource "aws_config_config_rule" "example" {
  name = "example-rule"

  source {
    owner             = "AWS"
    source_identifier = "S3_BUCKET_VERSIONING_ENABLED"
  }
}
```

This rule checks whether versioning is enabled on S3 buckets. By enforcing such rules, organizations can be certain they are adhering to data protection best practices.

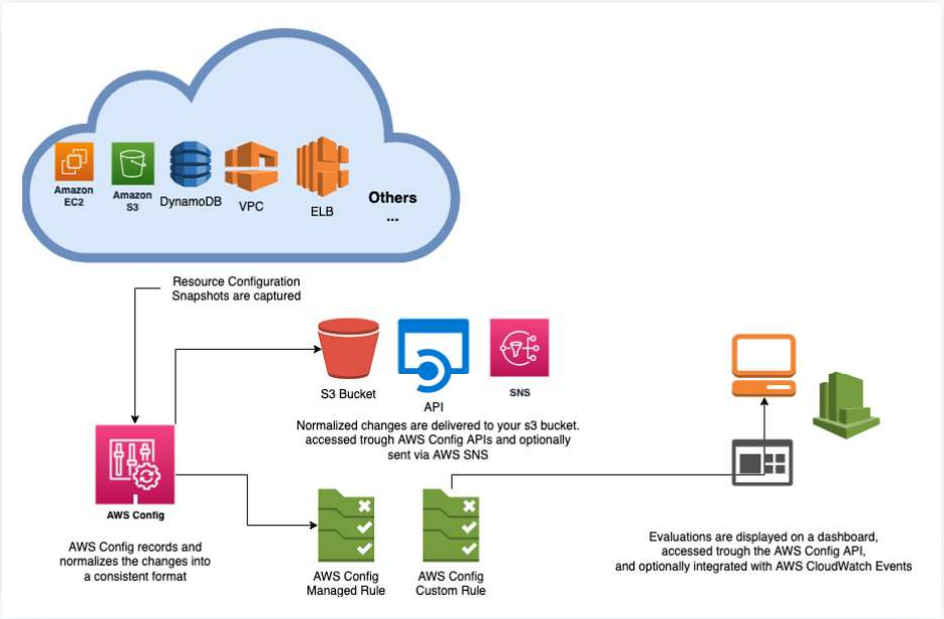


Figure 6: How AWS Config monitors resource configurations and compliance within an AWS environment (Source: AWS blog)

Securing networks

In today's cloud computing environment, companies must properly secure their networks to be sure their sensitive data is safe and their application's integrity is maintained. Amazon Web Services (AWS) offers tools and services for companies to create secure network architectures.

This section will discuss the key components involved in securing networks on AWS, focusing on Amazon Virtual Private Cloud (VPC), security groups, network ACLs, and AWS Web Application Firewall (WAF).

AWS VPC

An Amazon Virtual Private Cloud (VPC) enables the establishment of a logically isolated network in AWS, giving them complete control over their networking environment.

VPCs enable customization of network configurations, including IP address ranges, subnets, and route tables, ensuring security and organization.

Key features of AWS VPCs include:

- **Isolation:** VPCs create a private environment isolated from other AWS virtual networks, enhancing security.
- **Subnets:** Users can create public subnets for internet-accessible resources and private subnets for databases and application servers that are not exposed to external traffic.

Example Configuration of a VPC

Consider an organization looking to establish a VPC with a specified CIDR block. The following example illustrates how to create a VPC using infrastructure as code (IaC) with Terraform:

```
resource "aws_vpc" "example" {
  cidr_block           = "10.0.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true
}
```

This configuration defines a VPC with a CIDR block of `10.0.0.0/16`, enabling DNS support for instances within the VPC.

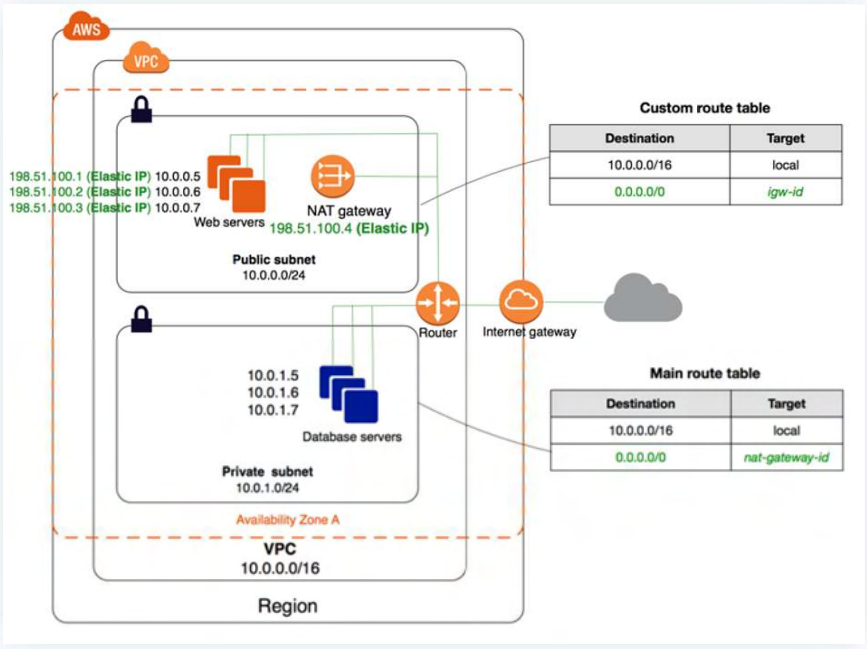


Figure 7: AWS VPC subnets and their connections to the internet and other AWS resources (Source: Stackoverflow.com)

Security groups and network ACLs

Security groups serve as virtual firewalls, monitoring inbound and outbound traffic to instances to safeguard your resources. Network access control lists (ACLs) stand guard at the subnet level; they reinforce security by allowing you to set rules that determine which traffic is allowed to enter or leave subnets.

Key features of the two include the following:

- **Security groups are stateful:** Responses to requests from a specified IP will are automatically permitted, no matter the outbound rules in place.
- **Network ACLs are stateless:** These enable enhanced traffic control at the subnet level, mandating that rules be established for both inbound and outbound traffic.

Example Configuration of security groups

In an organization that needs to allow HTTP traffic from any IP address, the following Terraform example illustrates the configuration of a security group:

```
resource “aws_security_group” “example” {
  vpc_id = aws_vpc.example.id

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = “tcp”
    cidr_blocks = [“0.0.0.0/0”]
  }
}
```

This security group allows HTTP traffic from any source IP address, facilitating access to web applications hosted on EC2 instances.

Example configuration of network ACLs

To complement the security group, a network ACL can be set up to allow inbound HTTP traffic and all outbound traffic:

```
resource “aws_network_acl” “example” {
  vpc_id = aws_vpc.example.id

  ingress {
    rule_no     = 100
    protocol   = “tcp”
    rule_action = “allow”
    cidr_block = “0.0.0.0/0”
    from_port  = 80
    to_port     = 80
  }

  egress {
    rule_no     = 100
    protocol   = “-1”
    rule_action = “allow”
    cidr_block = “0.0.0.0/0”
  }
}
```

This configuration demonstrates a network ACL that permits inbound HTTP traffic while allowing all outbound traffic.

Security Groups and Network ACLs

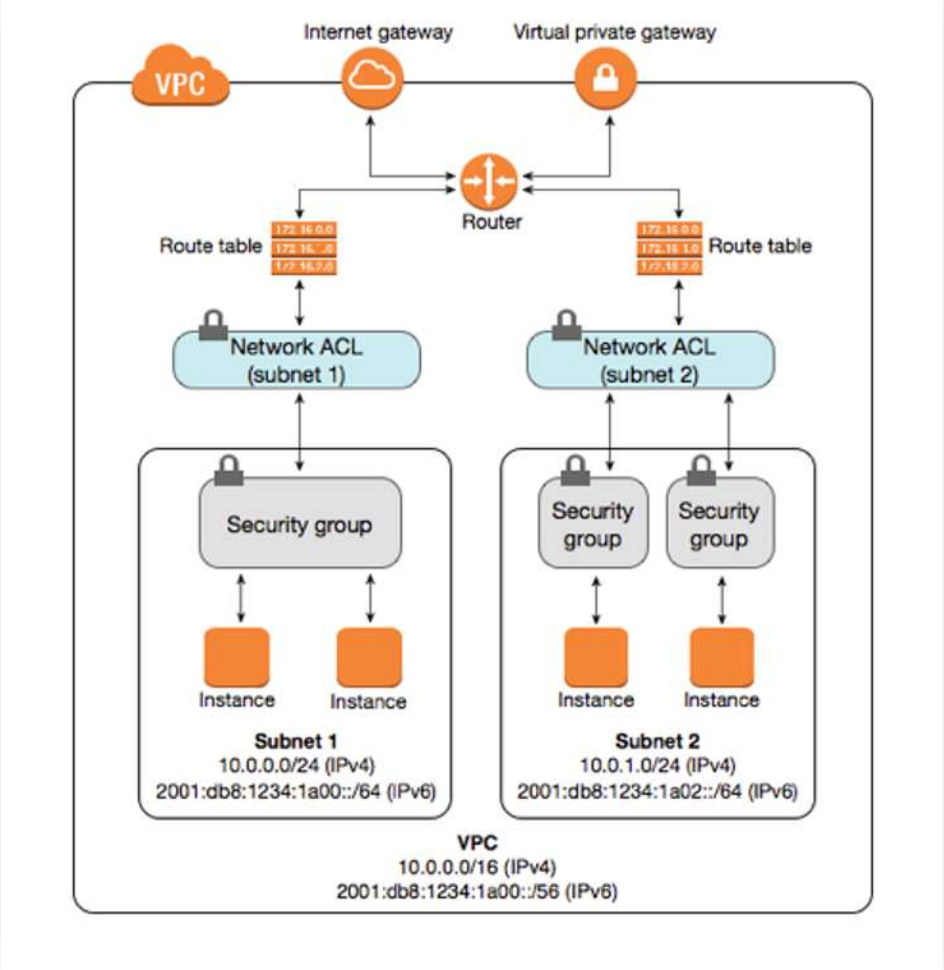


Figure 8: Role of security groups and network ACLs in controlling traffic within a VPC (Source: Medium blog)

AWS WAF

AWS Web Application Firewall (WAF) secures web apps from being compromised, including attacks that can impact availability or drain resources. By filtering and monitoring HTTP and HTTPS requests, AWS WAF enables organizations to create custom rules to protect their applications.

Key features of AWS WAF include:

- **Customizable rules:** Organizations can define rules according to various factors such as an HTTP header, URI string, or IP address.
- **Real-time monitoring:** AWS WAF gives visibility into traffic patterns and security threats as they occur.

Example Configuration of AWS WAF

In a scenario where an organization wants to create a WAF rule to filter requests based on specific criteria, the following example demonstrates how to set up an AWS WAF rule using Terraform:

```
resource "aws_waf_rule" "example" {
   name      = "example-waf-rule"
   metric_name = "ExampleWAFRule"

   predicates {
 data_id = aws_waf_byte_match_set.example.id
 negated = false
 type    = "ByteMatch"
 }
}
```

This configuration creates a WAF rule that matches requests based on predefined conditions, offering an essential layer of security for web applications.

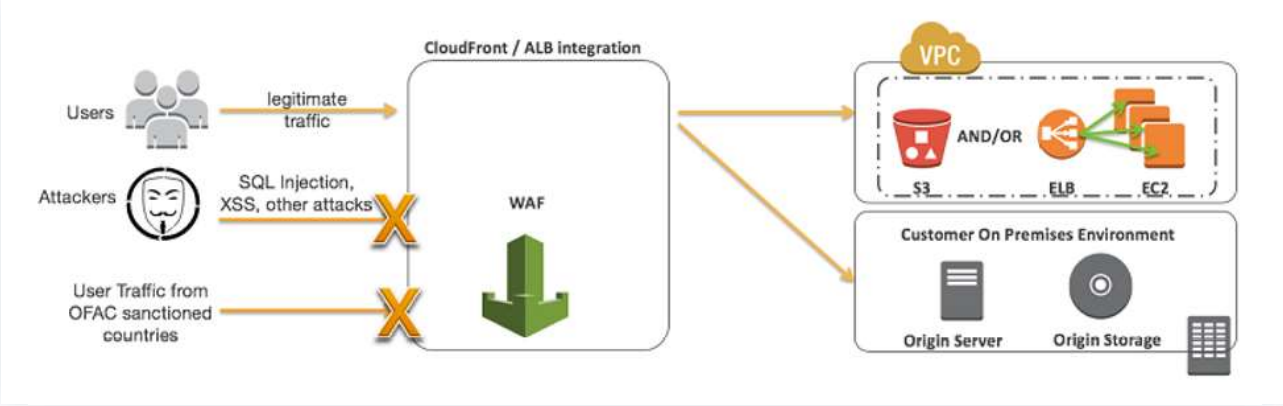


Figure 9: AWS WAF integration with other AWS services to protect web apps (Source: Bogotobogo.com)

For additional information, visit Amazon's AWS WAF page.

Protecting data

Data protection in the cloud is critical today, with new cyber threats emerging daily. Amazon Web Services (AWS) provides a range of tools to secure data, particularly through encryption and effective access controls.

This section explores key methods for protecting data in AWS, focusing on AWS Key Management Service (KMS) and S3 bucket policies.

Encryption with AWS KMS

AWS Key Management Service (KMS) allows users to create and manage keys for encrypting their data. Companies must manage encryption keys effectively to properly safeguard their sensitive information, prevent unauthorized access, and comply with regulatory requirements.

Key features of AWS KMS include:

- **Centralized key management:** AWS KMS provides a centralized location to manage encryption keys for all AWS services, simplifying the process of key administration.
- **Fine-grained access control:** You can define detailed permissions for who can use specific keys, allowing for a least-privilege approach to key management.
- **Automatic key rotation:** Enabling key rotation regularly enhances security by minimizing the risk associated with compromised keys.

Example configuration of AWS KMS

Consider a scenario where an organization needs to create a KMS key with automatic key rotation enabled. The following illustrates how to define a KMS key using IaC with Terraform:

```
resource “aws_kms_key” “example” {
  enable_key_rotation = true
  description         = “KMS key for encrypting sensitive data”
}
```

This configuration not only creates a KMS key but also enables automatic key rotation, which is a best practice for maintaining security over time.

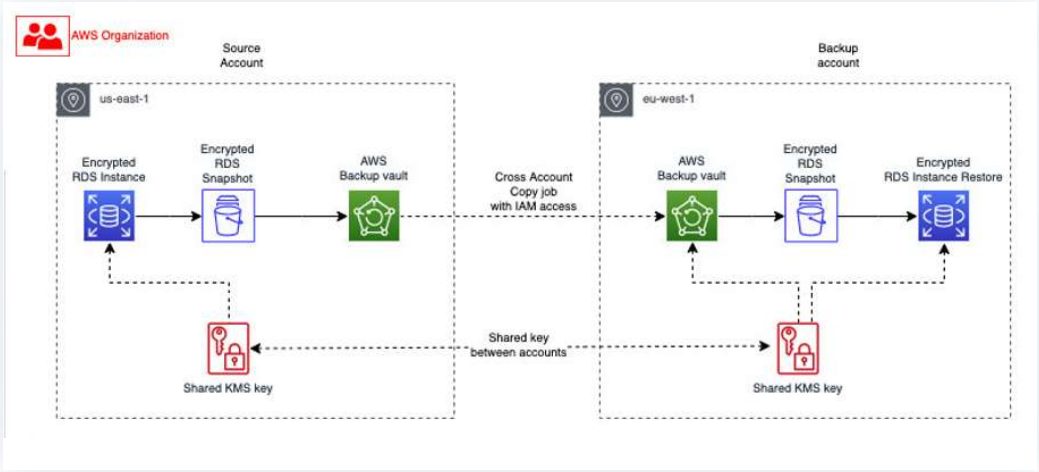


Figure 10: How AWS KMS integrates with other AWS services for encryption purposes (Source: AWS documentation)

For more information on AWS KMS, refer to the AWS KMS documentation.

S3 bucket policies

With Amazon S3 (Simple Storage Service), organizations can utilize bucket policies to enforce encryption settings and ensure that all data stored in S3 is encrypted. This will keep sensitive data secure and in compliance with data protection regulations.

Key features of S3 bucket policies include:

- **Access control:** S3 bucket policies define who can access data stored in S3 and what actions they can perform.
- **Mandatory encryption:** By specifying conditions in bucket policies, organizations can enforce encryption settings for all objects uploaded to a bucket.

Example S3 bucket policy for encryption

If a company wants to ensure that all data uploaded to a specific S3 bucket is encrypted using AWS KMS, the following example demonstrates the configuration of a bucket policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::example-bucket/*",
    "Condition": {
    "StringEquals": {
          "s3:x-amz-server-side-encryption": "aws:kms"
    }
    }
    }
  ]
}
```

In this policy, all uploads to the specified bucket must utilize AWS KMS encryption, meaning whatever data is stored in the bucket is automatically protected

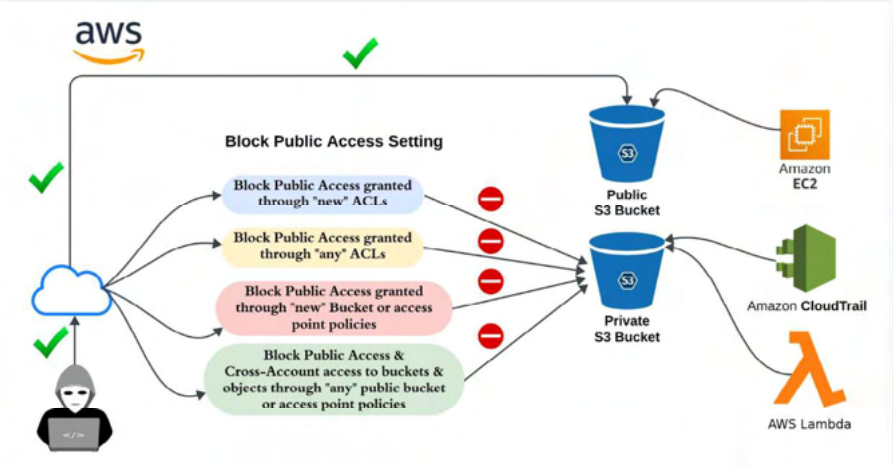


Figure 11: S3 bucket policy (Source: Qualys blog)

For additional details on S3 bucket policies, check out Amazon's documentation.

Automating incident response

The ability to respond swiftly to incidents today is crucial for mitigating risks and protecting sensitive information. AWS Lambda, a serverless compute service, empowers organizations to automate their incident response processes, ensuring that security events are managed efficiently and effectively.

AWS Lambda: A tool for automated incident response

AWS Lambda lets users execute code in response to events without managing server infrastructure; this enables rapid incident response and compliance with internal policies and regulations. This approach brings both scalability and cost-effectiveness; it also enjoys seamless integrations with various other AWS services such as CloudWatch and CloudTrail.

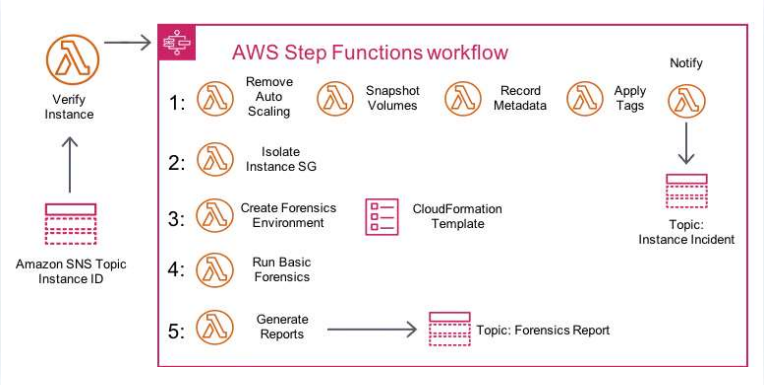


Fig 12: AWS Lambda incident response workflow (Source: Medium)

Example: automating incident response with AWS Lambda

In an example scenario, an organization may want to automatically respond to unauthorized access attempts by revoking a user's access. The following Python code snippet demonstrates how a Lambda function could be configured to handle such security events:

```
import json
import boto3

def lambda_handler(event, context):
      # Log the detected security event
      print("Security event detected:", event)

      # Initialize the IAM client
      iam = boto3.client('iam')

      # Logic to revoke user's access
      user_name = event['userName']

      try:
      # Remove the user's permissions (this is an example, adjust
as necessary)
  iam.detach_user_policy(
            UserName=user_name,
            PolicyArn='arn:aws:iam::aws:policy/YourPolicyName'
  )
  print(f"Access revoked for user: {user_name}")
      except Exception as e:
  print(f"Error revoking access: {str(e)}")

      return {
  'statusCode': 200,
  'body': json.dumps('Response executed!')
   }
```

In this example, the Lambda function logs the security event and then attempts to revoke access from a specified user by detaching their IAM policy. This action helps mitigate the risk posed by unauthorized access quickly and efficiently.

Integration with AWS Services

To enhance incident response capabilities, AWS Lambda can be triggered by various events, such as:

- **Amazon CloudWatch events:** Automate responses to security-related events, such as high CPU utilization or unauthorized API calls.
- **AWS CloudTrail:** Detect API call activities and initiate responses based on specific events, such as attempts to modify IAM policies.
- **Amazon SNS:** Send notifications to administrators or initiate workflows to address security incidents.

Enhancing security with third-party tools

In the rapidly evolving landscape of cloud security, integrating third-party tools such as AlgoSec's Prevasio is essential for organizations seeking to bolster their security posture.

Prevasio is a comprehensive AI-powered security management platform that offers advanced capabilities tailored for dynamic cloud environments, including AWS.

The platform delivers enhanced visibility and control over the security policies in force for both your cloud and on-premises environments. By leveraging Prevasio, companies can automate and optimize their security management processes, ensuring better compliance, reduced risk, and a more agile security framework.

Benefits of integrating AlgoSec Prevasio into your AWS environment

Advanced security management: The AlgoSec Prevasio security platform offers visibility into security policies across diverse environments, enabling organizations to optimize their security configurations and identify potential gaps.

Policy automation: The platform's automation features are designed to streamline the definition and enforcement of security policies, helping to minimize manual efforts and reduce human error, which enhances overall security response times.

Compliance and risk Management: Prevasio provides tools for managing compliance with regulations like GDPR and PCI DSS, including risk assessment and reporting capabilities that facilitate audits and compliance verification.

Importance of ongoing vigilance and adaptation

Continuous vigilance and adaptability are paramount in cloud security. As cyber threats evolve and regulatory requirements shift, organizations must commit to:

- **Regularly updating security policies:** Frequent reviews and updates to security policies ensure they remain effective against new vulnerabilities.
- **Proactive monitoring:** Continuous monitoring through tools like AWS CloudTrail and AWS Config is essential for detecting anomalies and ensuring compliance.
- **Training and awareness:** Making sure teams are aware of the latest security practices will raise their level of preparedness, lowering the chance of human error.
- **Leveraging automation:** Automating incident response and security management processes can drastically improve reaction times to threats and minimize manual errors.

By maintaining a proactive stance and regularly adapting to changes in the security landscape, organizations can better protect their cloud environments, mitigate risks, and ensure compliance, thereby safeguarding their critical assets in their AWS ecosystem.

About AlgoSec

AlgoSec, a global cybersecurity leader, empowers organizations to secure application connectivity by automating connectivity flows and security policy, anywhere.

The AlgoSec platform enables the world's most complex organizations to gain visibility, reduce risk, achieve compliance at the application-level and process changes at zero-touch across the hybrid network.

Over 1,800 of the world's leading organizations trust AlgoSec to help secure their most critical workloads across public cloud, private cloud, containers, and on-premises networks.

algosec

© AlgoSec Inc. All rights reserved. AlgoSec and the AlgoSec logo are registered trademarks of AlgoSec Inc. All other trademarks used herein are the property of their respective owners.

AlgoSec.com