SECURE IOWA CONFERENCE

October 6, 2021

Lessons Learned Deploying Modern Cloud Systems in Highly Regulated Environments

Eric Johnson

Principal Security Engineer, Puma Security

Senior Instructor, SANS Institute

www.linkedin.com/in/eric-m-johnson

@emjohn20

PUMA SECURITY

SESSION GOALS

- Review Department of Defense (DoD) Cloud Computing Security Requirements (SRG)
- Understand AWS organizations architecture and service control policies
- Build organization-wide VPC network security controls
- Centralize single tenant KMS encryption key management
- Ingest security service and custom vulnerabilities

COMPLIANCE VERSUS PUBLIC CLOUD

Security requirements and guidance for DoD and Cloud Service Providers (CSPs):

https://dl.dod.cyber.mil/wp-content/uploads/cloud/SRG/index.html

https://www.fedramp.gov/documents-templates/

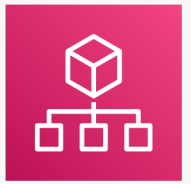
https://aws.amazon.com/compliance/services-in-scope/

SRG 5.10 Architecture

SRG 5.10: ARCHITECTURE

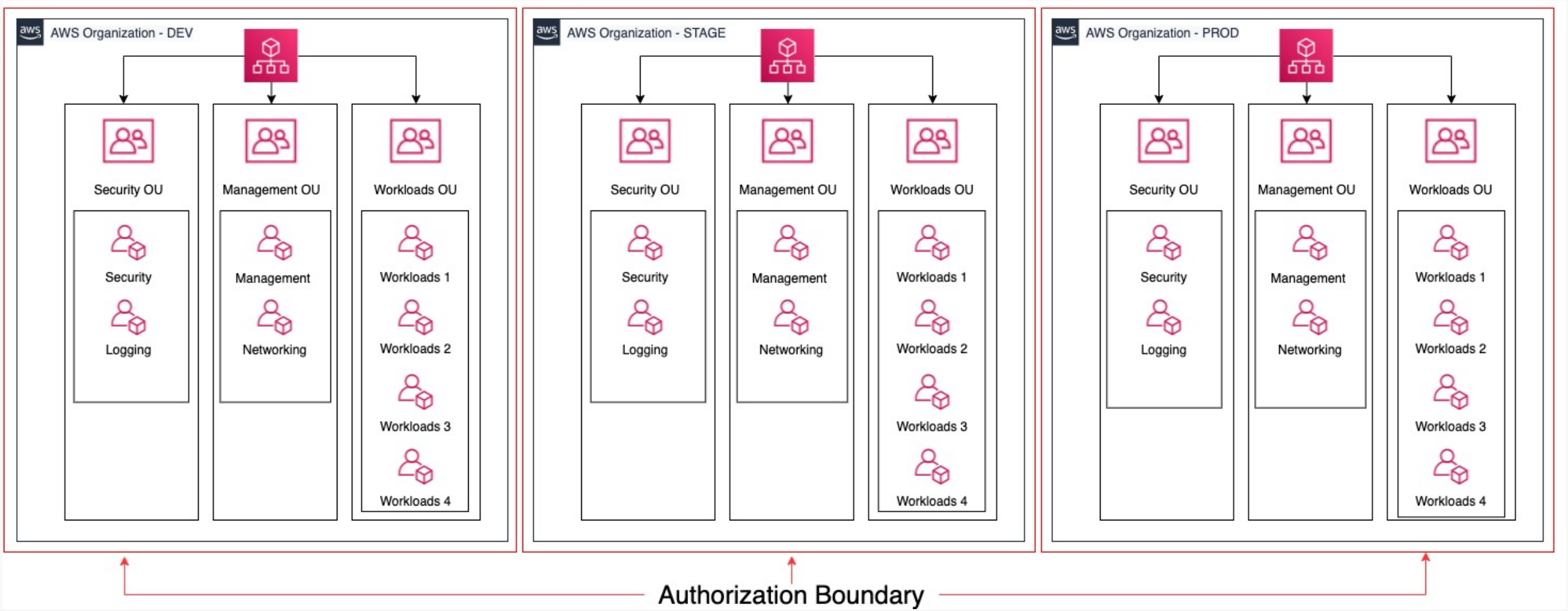
SRG 5.10 Architecture & Authorization Boundary Guidance controls:

- *NIST SP 800-37 defines an authorization boundary as "all components of an information system to be authorized for operation by an Authorizing Official (AO) and excludes separately authorized systems to which the information system is connected."*

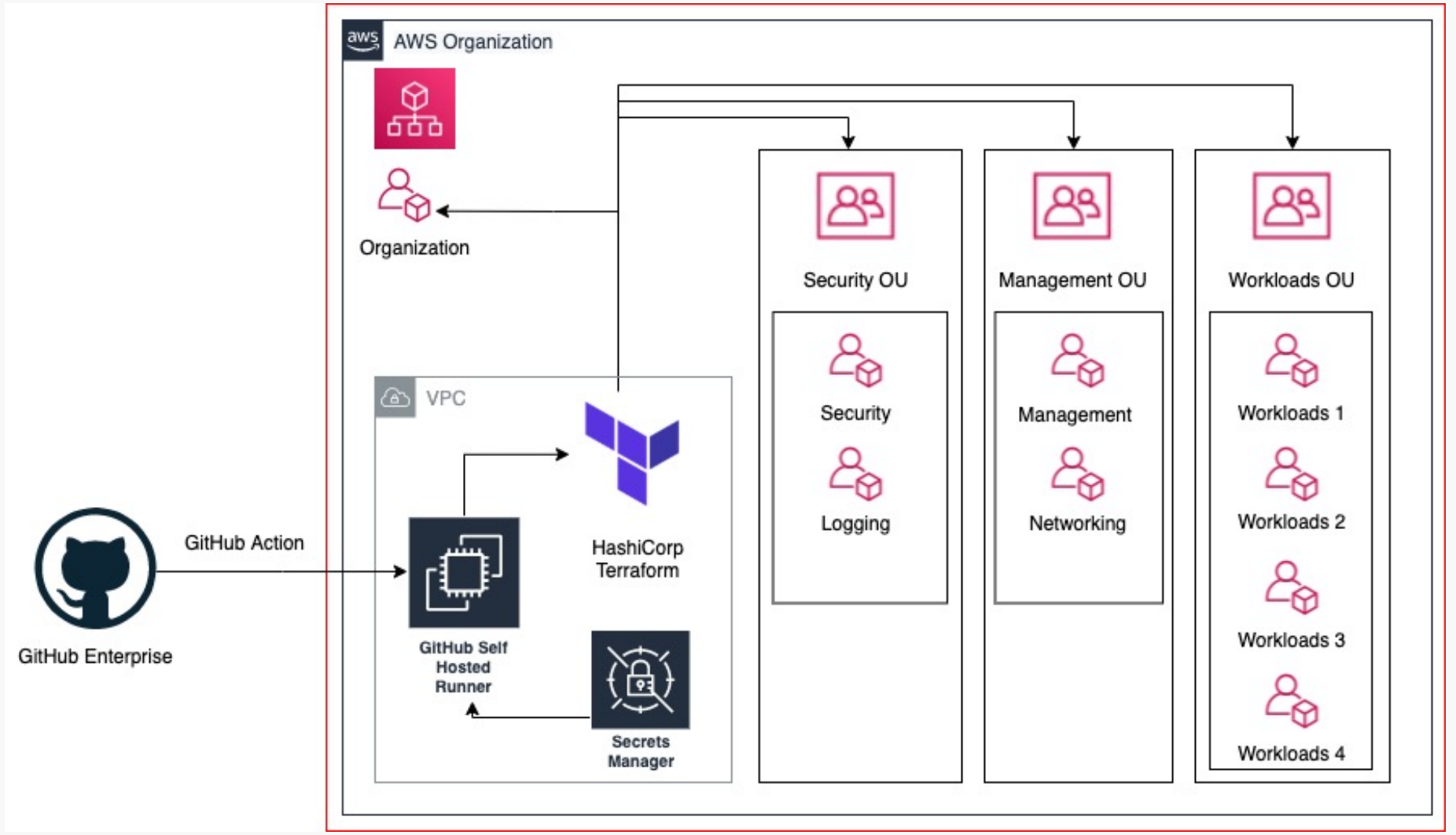


AWS Organizations

AWS ORGANIZATION AUTHORIZATION BOUNDARY



INFRASTRUCTURE AS CODE AUTHORIZATION BOUNDARY



FEDRAMP SC BASELINE REQUIREMENTS

AWS Service Control Policies provide guardrails for enforcing FedRAMP security controls across each organization:

1. AU-10 Non-repudiation – The information system protects against an individual falsely denying an action
2. AU-11: Audit Log Retention - Prevent the unauthorized deletion of log data

AWS Service Control Policy

AWS SCP RESOURCE: AU-10 NON-REPUDATION

Creating a Service Control Policy resource using Terraform:

```
resource "aws_organizations_policy" "monitoring_protection" {
  name        = "MonitoringProtection"
  description = "Block disabling logging and alerts."
  type        = "SERVICE_CONTROL_POLICY"

  content = data.aws_iam_policy_document.
            monitoring_protection_policy.json

  tags = var.tags
}
```

AWS Service Control Policy

AWS SCP POLICY: AU-10 NON-REPUDATION

SCP as Code: IAM Policy denying access to stop CloudTrail logging:

```
data "aws_iam_policy_document" "monitoring_protection_policy"
{
  statement {
    effect = "Deny"

    actions = [
      "cloudtrail:StopLogging",
    ]

    resources = ["*"]
}
```

AWS Service Control Policy

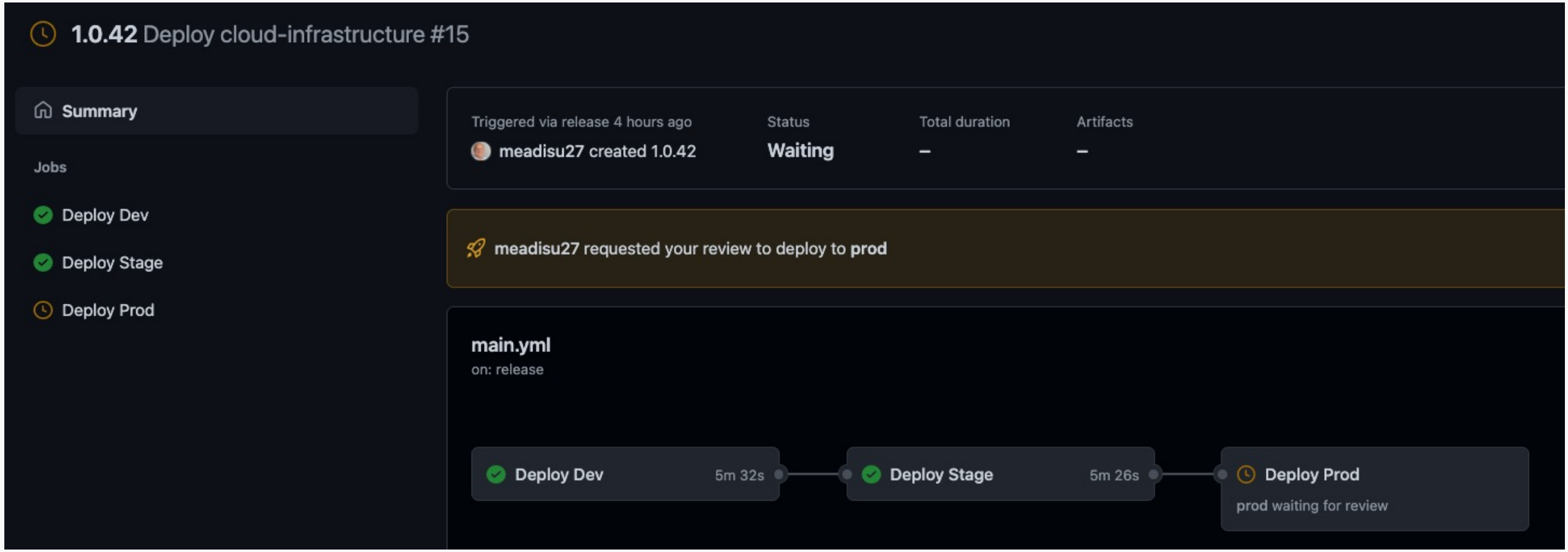
CREATING A GITHUB ACTION WORKFLOW

Configuring a workflow in the *.github/workflows/main.yml* file:

```
1  jobs:
2    deploy_prod:
3      name: Deploy Prod
4      environment:
5        name: prod
6        runs-on: [self-hosted, cloud-prod-01]
7      steps:
8        - name: Set version
9          run: |
10           echo "VERSION=${GITHUB_REF/refs\/tags\//}" >> $GITHUB_ENV
11       - uses: actions/checkout@v2
12       - name: Prod Terraform deployment
13         run: |
14           /bin/bash ./build/deploy.sh "prod"
```

DEPLOYING CLOUD INFRASTRUCTURE

Leveraging GitHub Actions to deploy AWS configuration:



SRG 5.10.1 Cloud Access Point (CAP)

SRG 5.10.1: CLOUD ACCESS POINT

SRG 5.10.1 Cloud Access Point (CAP):

☑ *Commercial cloud services used for Sensitive Data must be connected through a Cloud Access Point (CAP)*

AWS Transit Gateway

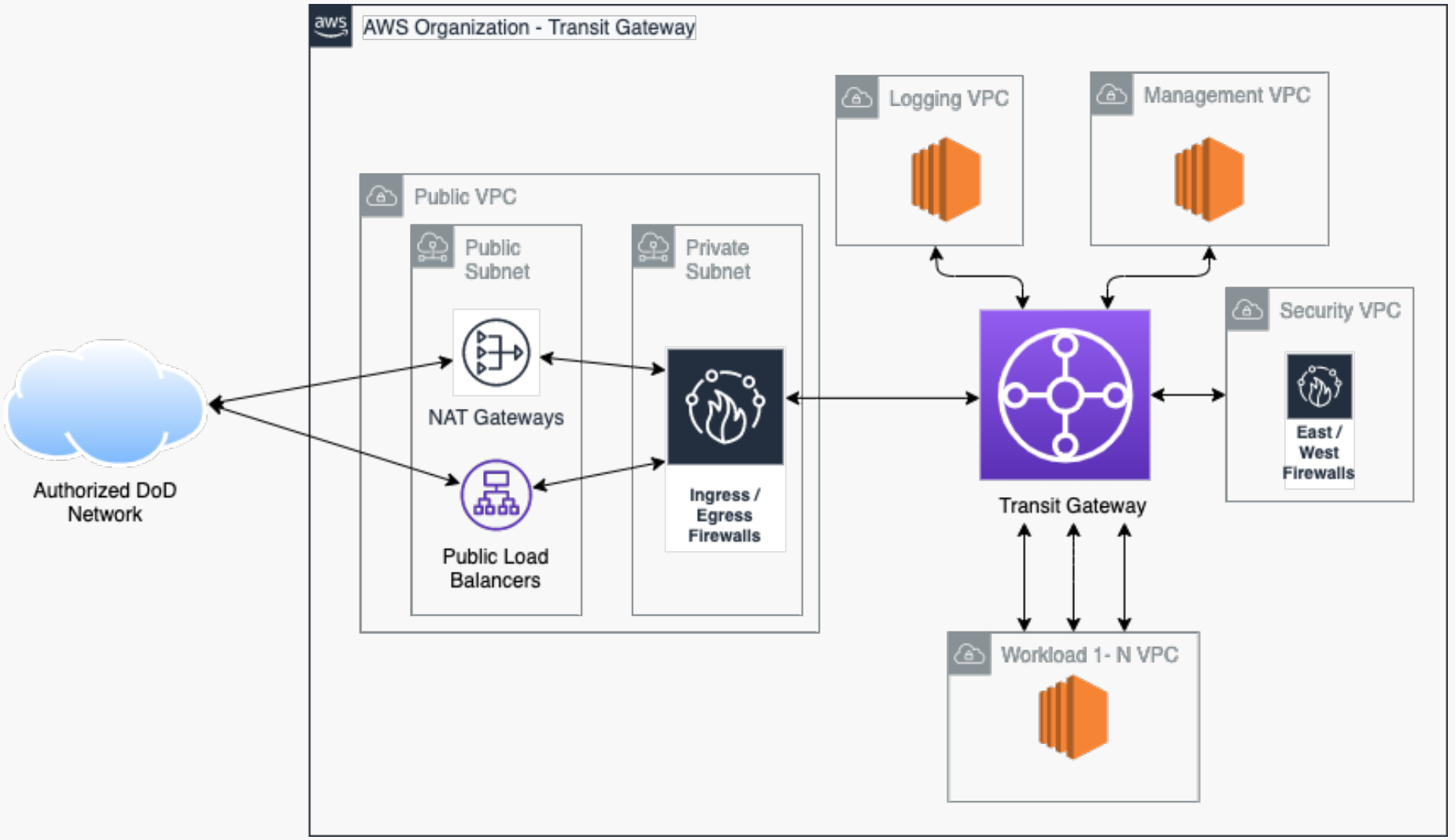
FedRAMP SC Baseline Requirements

System and Communications Protection FedRAMP security controls for organization ingress and egress traffic:

1. SC-7 (1): Monitors and controls communications at the external boundary of the system and at key internal boundaries within the system
2. SC-7 (3): Boundary Protection – The organization limits the number of external network connections
3. SC-7 (5): The information system at managed interfaces denies network communications traffic by default and allows network communications traffic by exception

AWS Transit Gateway

AWS ORGANIZATION TRANSIT GATEWAY NETWORK



AWS TRANSIT GATEWAY: TERRAFORM CONFIGURATION

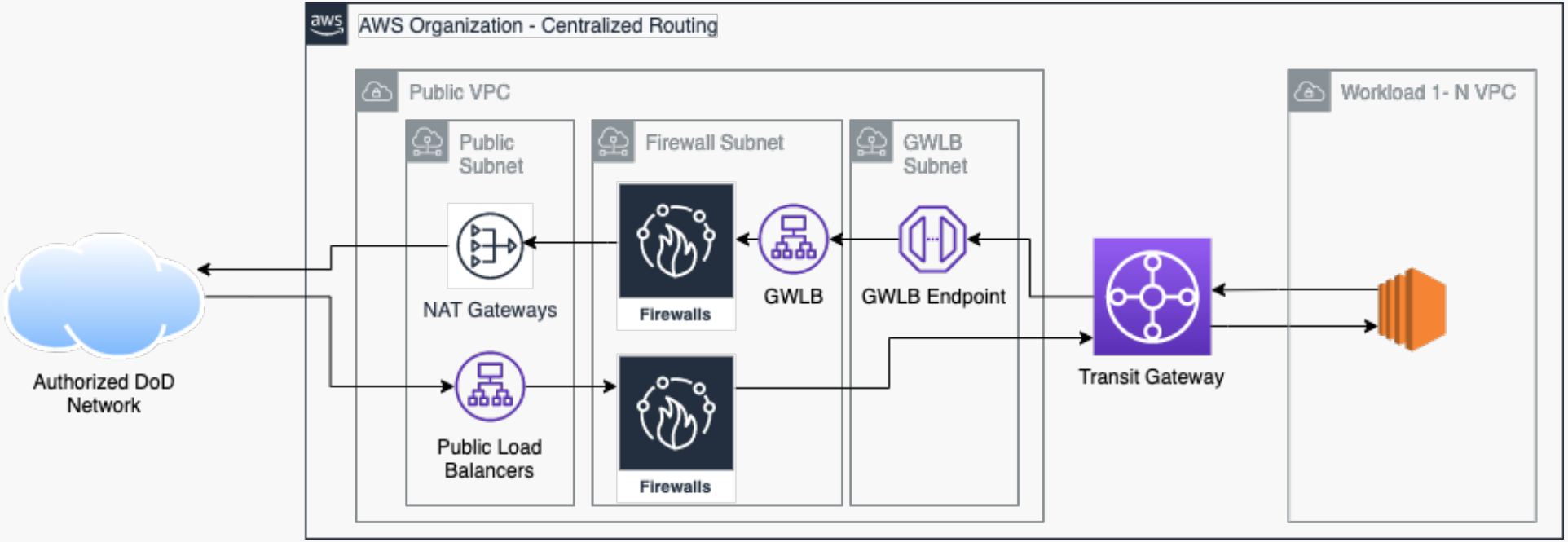
Creating a centralized TGW and sharing with the org accounts:

```
module "mgmt_tgw" {
  source  = "terraform-aws-modules/transit-gateway/aws"
  version = "2.5.0"

  name                                        = "network-${var.env}-tgw"
  amazon_side_asn                             = local.tgw_gateway_bgp_asn[var.env]
  enable_auto_accept_shared_attachments  = true

  share_tgw                          = true
  ram_name                           = "network-${var.env}-tgw-ram-share"
  ram_allow_external_principals = true
  ram_principals = [
    for account in var.organization_accounts :
    account.id if account.id != var.mgmt_account_id
  ]
}
```

AWS Transit Gateway

AWS ORGANIZATION CENTRALIZED TRAFFIC FLOW

- Ingress traffic enters from the authorized network to an external load balancer, through the firewalls, into the TGW, and to the workload
- Egress traffic leaves the workload, into the TGW, to a GWLB, through the firewalls, to the authorized network



AWS GWLB: TERRAFORM CONFIGURATION

Creating a centralized GWLB for egress traffic routing:

```
resource "aws_lb" "gwlb" {
  name               = "network-${var.env}-egress-gwlb"
  load_balancer_type = "gateway"

  subnets = [module.vpc.private_subnets["firewall-az-a"].id,
             module.vpc.private_subnets["firewall-az-b"].id,
             module.vpc.private_subnets["firewall-az-c"].id]
}

resource "aws_lb_target_group" "gwlb" {
  name     = "network-${var.env}-egress-gwlb-tg"
  port     = 6081
  protocol = "GENEVE"
  vpc_id   = module.vpc.vpc_id
  …
}
```

AWS Transit Gateway

SRG 5.11 Encryption of Data-at-Rest in Commercial Cloud Storage

SRG 5.11: CLOUD ENCRYPTION AT REST

AWS KMS CMK (customer managed keys) satisfy the core requirements (almost):

1. ✓ Uses FIPS 140-2 validated cryptography modules
2. ✓ Customer maintains control of the keys, from creation through storage and use to destruction
3. ✓ Encrypt all data at rest stored in virtual machine hard drives, block devices, database records, etc.
4. ✗ Enables high-assurance data spill remediation through cryptographic erasure and file deletion without the involvement or cooperation of a CSP

AWS Key Management Service

AWS KMS SCHEDULING KEY DELETION

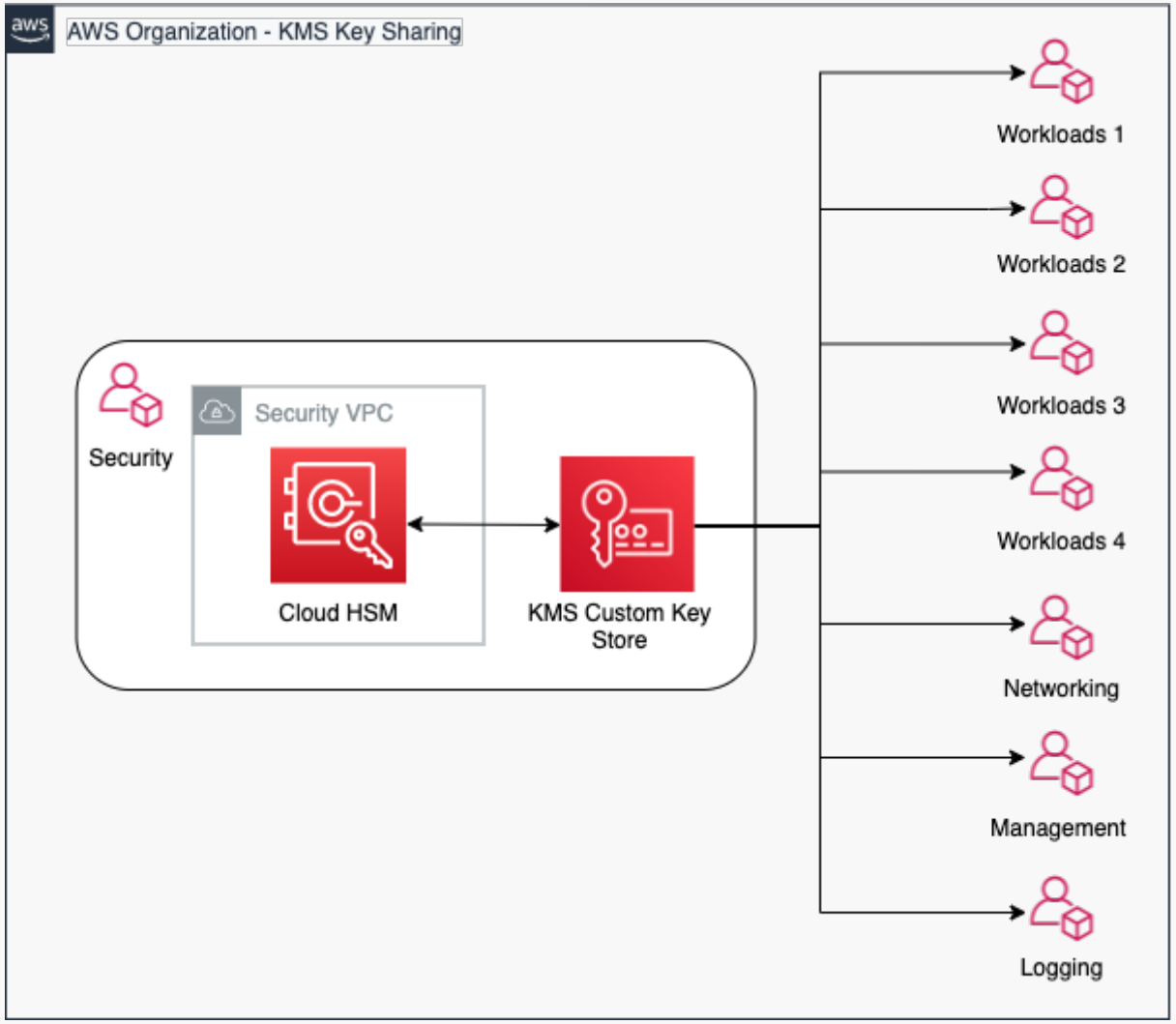
AWS KMS requires a scheduled waiting period before deleting the key material:

- Default waiting period is 30 days
- Minimum waiting period is 7 days
- Keys pending deletion cannot be used in cryptographic operations
- Keys pending deletion (and protected data) can be recovered within the waiting period

AWS Key Management Service

AWS CLOUD HSM: SINGLE TENANT HSM

- Cloud HSM clusters run in the Security VPC with HSMs deployed across availability zones for redundancy
- Initialization is required to configure certificates and cluster admin password
- Activation requires a *kmsuser* for KMS custom key store integration
- New KMS CMKs are created using an external origin *AWS_CLOUDHSM* for the key material



AWS CLOUD HSM: TERRAFORM CONFIGURATION

AWS CloudHSM

Creating a Cloud HSM Cluster and HSM resource:

```
1  resource "aws_cloudhsm_v2_cluster" "cluster" {
2    hsm_type   = "hsm1.medium"
3    subnet_ids = [module.vpc.private_subnets["hsm-az-a"].id,
4                  module.vpc.private_subnets["hsm-az-b"].id,
5                  module.vpc.private_subnets["hsm-az-c"].id
6    ]
7  }
8
9  resource "aws_cloudhsm_v2_hsm" "hsm_a" {
10   subnet_id = module.vpc.private_subnets["hsm-az-a"].id
11   cluster_id = aws_cloudhsm_v2_cluster.cluster.cluster_id
12 }
13
```

AWS CLOUD HSM: CLUSTER ACTIVATION

Cloud HSM administrators must sign into the HSM, initialize, and activate the cluster before connecting the custom key store:

```
# Initialize the cluster
aws cloudhsmv2 initialize-cluster --cluster-id ${clusterid} \
--signed-cert CustomerHsmCertificate.crt \
--trust-anchor customerCA.crt

# Sign in to HSM, change PRECO password, and create kmsuser

# Create KMS custom keystore
aws kms create-custom-key-store --custom-key-store-name ${custom_keystore_name} \
--cloud-hsm-cluster-id ${clusterid} --key-store-password ${kms_password} \
--trust-anchor-certificate file://customerCA.crt


```

AWS CloudHSM

AWS KMS CMK: CROSS-ACCOUNT PERMISSIONS

AWS Key Management Service

Key policy enabling org cross-account usage:

```
statement {
  sid = "AllowOrgAccounts"
  effect = "Allow"

  principals {
    type = "AWS"
    identifiers = [
      for account in var.organization_accounts :
      "arn:${var.prefix}:iam::${account.id}:root"
      if account.id != var.sec_account_id
    ]
  }

  actions = ["kms:Decrypt","kms:DescribeKey","kms:Encrypt","kms:GenerateDataKey*",
             "kms:ReEncrypt*","kms:CreateGrant","kms:ListGrants","kms:RevokeGrant"
  ]
```

SRG 5.3 Ongoing Assessment

&

SRG 5.17 Supply Chain Risk Management Assessment

SRG 5.3 & SRG 5.17 SECURITY CONTROLS

Organization developer security, supply chain, and FedRAMP security controls for organization ingress and egress traffic:

1. SA-11 (1): The organization requires the developer of the information system, system component, or information system service to employ static code analysis tools to identify common flaws and document the results of the analysis.
2. SA-12: The organization protects against supply chain threats to the information system, system component, or information system service

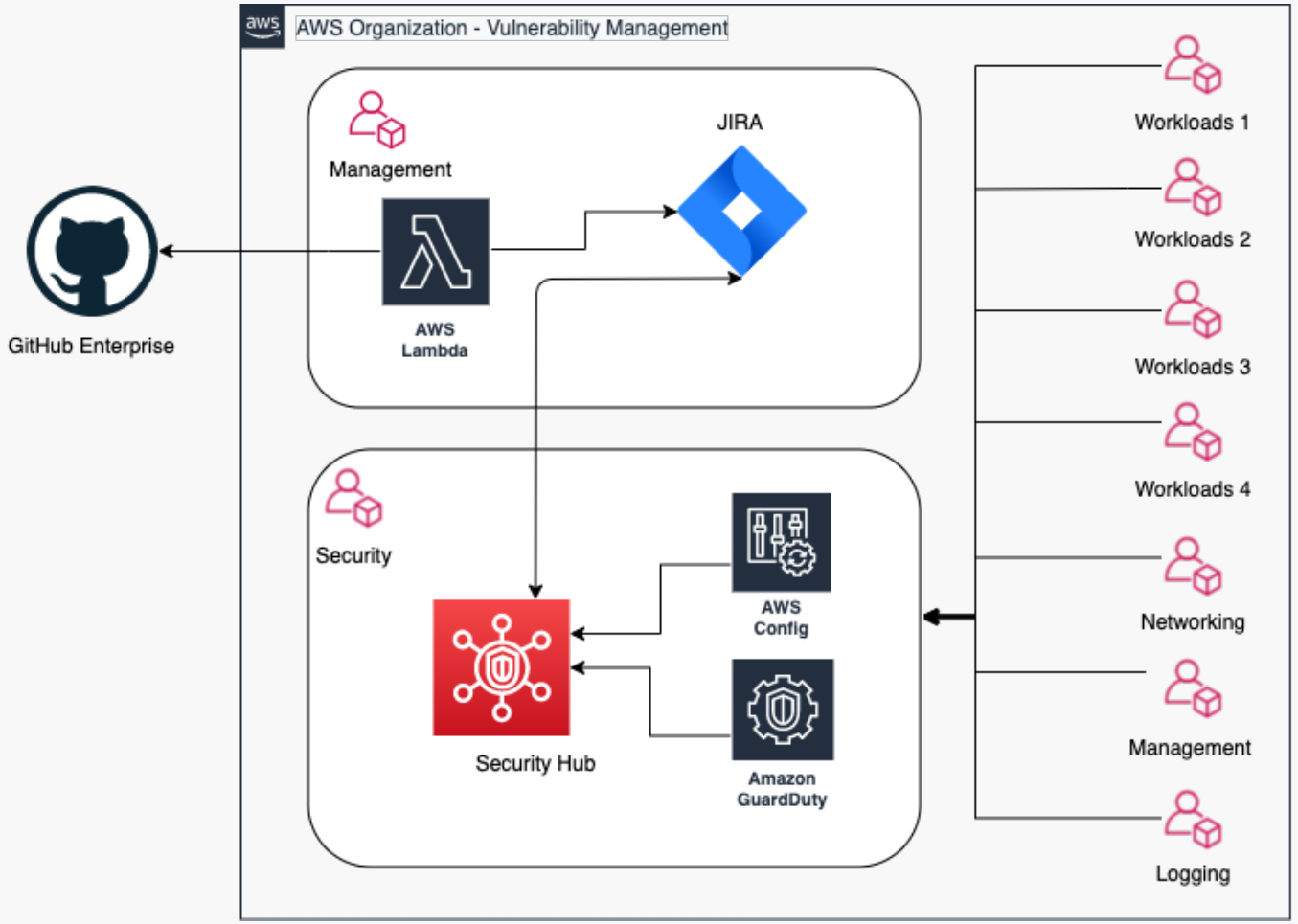
SRG 5.3 & SRG 5.17 SECURITY CONTROLS

Organization developer security, supply chain, and FedRAMP security controls for organization ingress and egress traffic:

3. RA-5 (3): The organization employs vulnerability scanning procedures that can identify the breadth and depth of coverage
4. RA-5 (10): The organization correlates the output from vulnerability scanning tools to determine the presence of multi-vulnerability/multi-hop attack vectors.

VULNERABILITY MANAGEMENT: AWS ORGANIZATION CENTRALIZED

- Security account is the delegated administrator for Security Hub / Guard Duty
- Security account aggregates config data for the organization
- Security Hub findings sync to JIRA using the **AWS Service Management Connector** for JSM
- GitHub code scanning JIRA integration runs in Lambda to pull GH Advanced Security alerts



AWS SECURITY HUB: TERRAFORM CONFIGURATION

Delegating the security account as the Security Hub administrator:

```
1  resource "aws_organizations_organization" "this" {
2    aws_service_access_principals = [
3      "guardduty.amazonaws.com",
4      "securityhub.amazonaws.com",]
5    feature_set                   = "ALL"
6  }
7
8  resource "aws_securityhub_organization_admin_account" "this" {
9    depends_on = [aws_organizations_organization.this]
10
11   admin_account_id = var.sec_account_id
12 }
13
14 resource "aws_securityhub_organization_configuration" "this" {
15   auto_enable = true
16 }
```

AWS Security Hub

AWS GUARD DUTY: TERRAFORM CONFIGURATION

Delegating the security account as the Guard Duty administrator:

```
resource "aws_guardduty_organization_admin_account" "default" {
  admin_account_id = var.sec_account_id
}

resource "aws_guardduty_member" "member" {
  for_each = {
    for account in var.organization_accounts : account.id => account
    if account.id != var.admin_account_id
  }

  detector_id = var.guardduty_detector_id
  account_id  = each.value.id
  email       = each.value.email
}
```

Amazon GuardDuty

AWS CONFIG: TERRAFORM CONFIGURATION

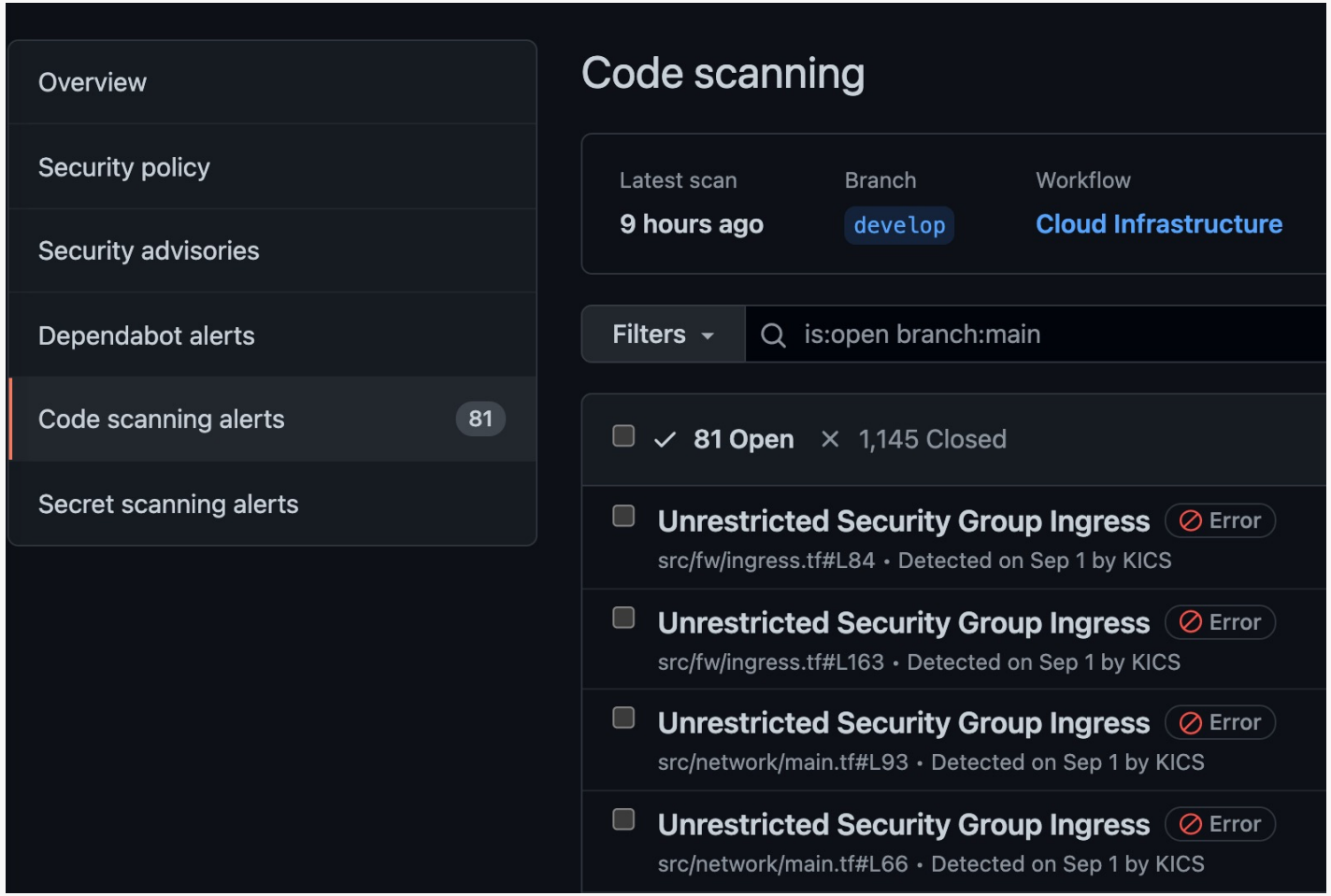
Delegating the security account as the Config aggregator:

```
resource "aws_organizations_delegated_administrator" "this"
{
  account_id = var.sec_account_id
  service_principal = "config.amazonaws.com"
}

resource "aws_config_configuration_aggregator" "this" {
  name = "org-${var.env}-config-aggregator"

  account_aggregation_source {
    account_ids = [
      for account in var.organization_accounts :
      account.id]
    regions = [var.region]
  }
}
```

VULNERABILITY MANAGEMENT: GITHUB SECURITY

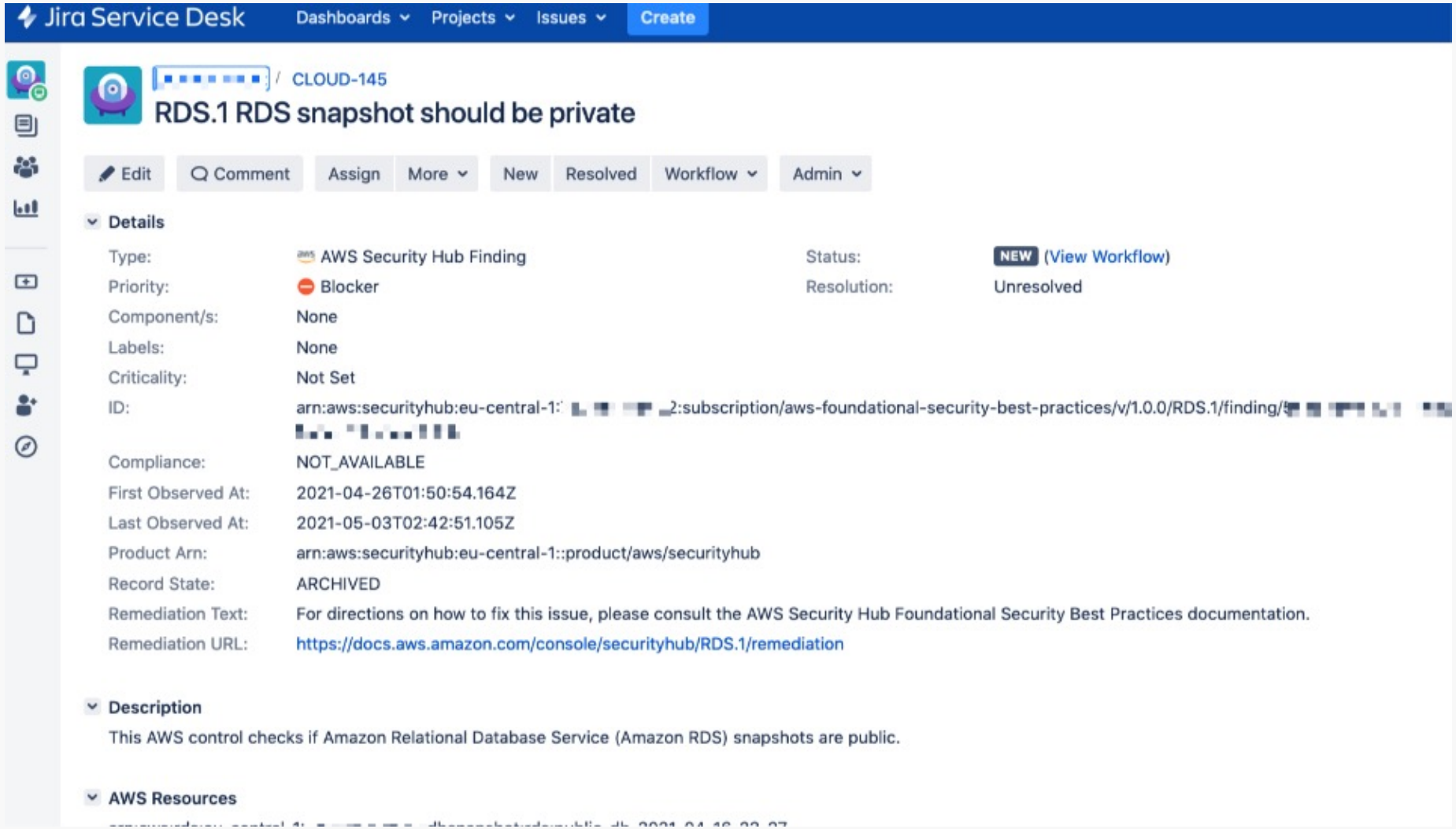
GitHub Advanced Security supports:

- Supply chain scanning and Dependabot alerts
- Code scanning with CodeQL
- Secret scanning alerts
- Alerts integration with JIRA



https://github.com/github/codescanning-jira-integration

JIRA SERVICE DESK: CENTRALIZED VULNERABILITY MANAGEMENT



ACKNOWLEDGEMENTS

- Marc Baker – Information Security Manager
- Eric Mead – Principal Security Engineer, Puma Security
- Allen Gonzalez – Systems Network Engineer

SECURE IOWA CONFERENCE

October 6, 2021

THANK YOU FOR ATTENDING!

QUESTIONS?

Eric Johnson

Principal Security Engineer, Puma Security

Senior Instructor, SANS Institute

www.linkedin.com/in/eric-m-johnson

@emjohn20

PUMA SECURITY