Summary

We will use AWS Transfer for SFTP to enable SFTP-based uploads to S3 buckets from different agencies. Using this service, organizations can submit data straight into S3 via SFTP. Terraform will be used to automate the solution in order to guarantee reproducibility, manageability, and scalability. Along with adhering to best practices for security and cost effectiveness, it will also provide monitoring and alerting capabilities for data upload errors.

Components of Architecture

1. AWS Transfer: Managed SFTP service to enable safe uploads of data.
2. File upload storage with encryption and appropriate access control is provided via Amazon S3.
3. AWS Lambda: To manage upload oversight and notifications.
4. Lambda function triggers and S3 buckets are monitored by Amazon CloudWatch.
5. Simple Notification Service (Amazon SNS): For informing the SRE team of updates.
6. Using Terraform, AWS resource deployment and management can be automated.

Detailed Resolution

1. **AWS Transfer for Configuring SFTP:**
   1. Establish an AWS Transfer for the EU-West-1 region's SFTP server.
   2. To separate the data for each agency, configure user-specific directories in the S3 bucket.
   3. Give each SFTP user least privilege access to the directories they have been assigned by using IAM roles.

2. **Configuring an S3 bucket:**
   1. Create S3 buckets with SSE-S3 or SSE-KMS server-side encryption enabled.
   2. Use bucket policies to limit access to SFTP users and specific IAM roles only.
   3. To save storage expenses, enable versioning and lifecycle controls.

3. **Roles and Policies of IAM:**
   1. Establish IAM roles with policies that provide the bare minimum of access rights.
   2. To maintain IAM roles and make sure the least privilege principle is followed, use Terraform.

4. **Automation for Onboarding and Offboarding:**
   1. Create Terraform scripts that configure IAM roles and policies and allow for the creation and deletion of SFTP users.
   2. Verify that scripts can do the task within the SLA of six hours.

5. **Observation and Warning:**
   1. To keep an eye on S3 bucket activities, use CloudWatch.
   2. Configure a CloudWatch Event Rule to start an AWS Lambda function in the event that no files are uploaded in the anticipated amount of time.
   3. The Lambda function notifies the SRE team via email or Slack by sending notifications via SNS.

6. **Terraform's Infrastructure as Code:**
   1. Divide the Terraform code into modules for configuring S3 buckets, IAM roles, rules, and SFTP setup.
   2. Use version control for the Terraform code at GitHub, where it has an organized repository and detailed documentation.

7. **Cost-Reduction Strategy:**
   1. When feasible, use the AWS Free Tier.
   2. Use S3 lifecycle policies to move legacy data to S3 Glacier, a less expensive storage class.
   3. Track AWS expenses and create alerts for your budget.

Example Terraform Code Structure

Bash

Directory Structure

```
/terraform
  /modules
    /sftp
      - main.tf
      - outputs.tf
      - variables.tf
    /s3
      - main.tf
      - outputs.tf
      - variables.tf
    /iam
      - main.tf
      - outputs.tf
      - variables.tf
  - main.tf
  - variables.tf
  - outputs.tf
  - providers.tf
```

- terraform.tfvars

/docs

- README.md
- SECURITY.md
- ONBOARDING.md
- OFFBOARDING.md

/scripts

- onboard_agency.sh
- offboard_agency.sh
- monitor_uploads.sh

/.github

- workflows
  - ci.yml
  - cd.yml

Sample Terraform Code

SFTP Module

Hcl

```
// modules/sftp/main.tf
resource "aws_transfer_server" "sftp" {
  identity_provider_type = "SERVICE_MANAGED"
  region                 = var.region
}
resource "aws_transfer_user" "user" {
  count        = length(var.agencies)
  server_id    = aws_transfer_server.sftp.id
  user_name    = var.agencies[count.index].name
```

```
  home_directory = "/${var.agencies[count.index].name}"
  role {
    arn = aws_iam_role.sftp_role.arn
  }

  tags = {
    Name = var.agencies[count.index].name
  }
}
```

S3 Module

Hcl

```
// modules/s3/main.tf
resource "aws_s3_bucket" "data_lake" {
  bucket = var.bucket_name

  versioning {
    enabled = true
  }

  server_side_encryption_configuration {
    rule {
      apply_server_side_encryption_by_default {
        sse_algorithm = "AES256"
      }
    }
  }

  lifecycle_rule {
```

```
    id      = "move-to-glacier"
    enabled = true

    transition {
      days          = 30
      storage_class = "GLACIER"
    }
  }
}
```

IAM Module

Hcl

```
// modules/iam/main.tf
resource "aws_iam_role" "sftp_role" {
  name = "sftp-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Effect = "Allow",
      Principal = {
        Service = "transfer.amazonaws.com"
      }
      Action = "sts:AssumeRole"
    }]
  })
}
resource "aws_iam_policy" "sftp_policy" {
  name   = "sftp-policy"
```

```
  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Effect = "Allow",
      Action = ["s3:PutObject", "s3:ListBucket"],
      Resource = ["arn:aws:s3:::${var.bucket_name}/*"]
    }]
  })
}
resource "aws_iam_role_policy_attachment" "sftp_policy_attachment" {
  role       = aws_iam_role.sftp_role.name
  policy_arn = aws_iam_policy.sftp_policy.arn
}
```

GitHub Repository Structure

- README.md: Overview of the project, setup instructions, and usage guidelines.
- SECURITY.md: Details on security measures, IAM roles, and policies.
- ONBOARDING.md: Steps for onboarding a new agency.
- OFFBOARDING.md: Steps for offboarding an agency.
- Terraform Modules: Organized in the terraform/modules directory.
- CI/CD Workflows: Automated workflows for continuous integration and deployment in the .github/workflows directory.

Summary

This solution gives agencies a scalable, safe, and affordable way to transfer data to S3 via SFTP by utilizing AWS services and infrastructure as code concepts. The solution complies with best practices for security and cost management, ensuring easy maintenance and scalability, by automating the deployment and management with Terraform.