Phase 1 실행 가이드

📚 학습 목표

Phase 1에서 배울 내용:

1. Terraform 기본 구조 이해
2. AWS Provider 설정
3. 모듈 시스템 사용법
4. VPC와 Security Group 생성
5. IAM Role과 Policy 관리
6. S3 Bucket 생성 및 설정

📂 Phase 1 파일 구조

```
phase1-code/
├── modules/
│   ├── network/          # VPC, Security Group
│   │   ├── main.tf       # 리소스 정의
│   │   ├── variables.tf  # 입력 변수
│   │   └── outputs.tf    # 출력 값
│   ├── iam/              # IAM Roles
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   └── outputs.tf
│   └── storage/          # S3 Bucket
│       ├── main.tf
│       ├── variables.tf
│       └── outputs.tf
└── environments/
    └── dev/              # 개발 환경
        ├── main.tf           # 모듈 호출
        ├── variables.tf      # 변수 정의
        ├── terraform.tfvars  # 실제 값 설정
        └── outputs.tf        # 출력 정의
```

🎯 코드 학습 포인트

1. main.tf 구조

```hcl
```

```hcl
terraform {
  required_version = ">= 1.0"    # Terraform 버전
  required_providers {           # 사용할 Provider
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.region            # 변수 사용
}

module "network" {               # 모듈 호출
  source = "../../modules/network"

  project_name = var.project_name # 변수 전달
  environment  = var.environment
}
```

학습 포인트:

- `terraform {}` 블록: 버전 및 Provider 설정
- `provider {}` 블록: AWS 리전 등 설정
- `module {}` 블록: 재사용 가능한 모듈 호출
- `var.xxx`: 변수 참조

2. variables.tf 구조

```hcl
variable "project_name" {
  description = "프로젝트 이름"  # 설명
  type        = string         # 타입
}

variable "archive_retention_days" {
  description = "보관 기간"
  type        = number
  default     = 7              # 기본값
}
```

학습 포인트:

- `description`: 변수 설명 (문서화)

- `type`: 변수 타입 (string, number, bool, list, map 등)
- `default`: 기본값 (선택사항)

3. outputs.tf 구조

```hcl
output "vpc_id" {
  description = "VPC ID"
  value       = module.network.vpc_id
}
```

학습 포인트:

- 다른 모듈이나 사용자에게 값 전달
- `terraform output` 명령으로 확인 가능
- 모듈 간 데이터 전달에 사용

4. 리소스 정의 (main.tf 내부)

```hcl
resource "aws_s3_bucket" "archive" {
  bucket = "${var.project_name}-archive"

  tags = {
    Name = "My Archive"
  }
}
```

학습 포인트:

- `resource`: AWS 리소스 생성
- 첫 번째 인자: 리소스 타입
- 두 번째 인자: 로컬 이름
- `${}`: 문자열 보간

5. Data Source 사용

```hcl
```

```
data "aws_vpc" "default" {
  default = true
}

# 사용
vpc_id = data.aws_vpc.default.id
```

학습 포인트:

- `data`: 기존 리소스 조회
- 생성하지 않고 읽기만 함
- 기본 VPC 같은 기존 리소스 활용

🚀 실행 전 준비

1. AWS 인증 설정 확인

bash

```
# AWS CLI 설치 확인
aws --version

# 인증 정보 확인
aws sts get-caller-identity

# 출력 예시:
# {
#     "UserId": "AIDACKCEVSQ6C2EXAMPLE",
#     "Account": "123456789012",
#     "Arn": "arn:aws:iam::123456789012:user/your-name"
# }
```

인증되지 않은 경우:

bash

```
aws configure
# AWS Access Key ID: 입력
# AWS Secret Access Key: 입력
# Default region: ap-northeast-2
# Default output format: json
```

2. Terraform 설치 확인

bash

```
terraform version

# 출력 예시:
# Terraform v1.6.0
```

3. 코드 다운로드

```
bash

# phase1-code 폴더 다운로드
# (이미 받으셨다면 스킵)

cd phase1-code/environments/dev
```

📝 실행 단계

Step 1: terraform.tfvars 수정

```
bash

# 파일 열기
nano terraform.tfvars
# 또는
code terraform.tfvars
```

수정할 내용:

```
hcl

project_name = "my-live-stream"    # 원하는 이름으로 변경
environment  = "dev"
region       = "ap-northeast-2"

# Owner 태그 수정
tags = {
  Project     = "LiveStreaming"
  Environment = "Development"
  ManagedBy   = "Terraform"
  Owner       = "YourName"        # 여기에 본인 이름 입력
  Purpose     = "Portfolio"
}
```

Step 2: Terraform 초기화

```
bash
```

```
cd environments/dev

# Terraform 초기화 (Provider 다운로드)
terraform init
```

출력 예시:

```
Initializing modules...
Initializing the backend...
Initializing provider plugins...
- Installing hashicorp/aws v5.x.x...

Terraform has been successfully initialized!
```

이 단계에서 하는 일:

- `.terraform/` 폴더 생성
- AWS Provider 플러그인 다운로드
- Backend 초기화
- 모듈 다운로드

Step 3: 실행 계획 확인

```
bash

# 어떤 리소스가 생성될지 미리 확인
terraform plan
```

출력 분석:

```
Terraform will perform the following actions:

  # module.storage.aws_s3_bucket.archive will be created
  + resource "aws_s3_bucket" "archive" {
      + bucket = "my-live-stream-dev-archive-123456789012"
      ...
    }

  # module.network.aws_security_group.medialive_input will be created
  + resource "aws_security_group" "medialive_input" {
      + name = "my-live-stream-dev-medialive-input"
      ...
    }

  # module.iam.aws_iam_role.medialive will be created
  + resource "aws_iam_role" "medialive" {
      + name = "my-live-stream-dev-medialive-role"
      ...
    }

Plan: 15 to add, 0 to change, 0 to destroy.
```

확인 사항:

- `Plan: X to add`: X개 리소스 생성
- `0 to change`: 기존 리소스 변경 없음
- `0 to destroy`: 삭제할 리소스 없음

Step 4: 리소스 생성

```
bash

# 실제로 AWS에 리소스 생성
terraform apply
```

프롬프트:

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

'yes' 입력 후 Enter

생성 과정:

```
module.storage.aws_s3_bucket.archive: Creating...
module.network.aws_security_group.medialive_input: Creating...
module.iam.aws_iam_role.medialive: Creating...
...
Apply complete! Resources: 15 added, 0 changed, 0 destroyed.
```

소요 시간: 약 1~2분

Step 5: 결과 확인

```bash
# 모든 출력 확인
terraform output

# 특정 출력만 확인
terraform output vpc_id
terraform output archive_bucket_name

# 완료 메시지 확인
terraform output phase1_completion
```

✅ 생성된 리소스 확인

AWS Console에서 확인

1. **S3 Bucket**

```
AWS Console → S3
→ 버킷 이름: my-live-stream-dev-archive-xxxxx
```

2. **Security Group**

```
AWS Console → EC2 → Security Groups
→ 이름: my-live-stream-dev-medialive-input
```

3. **IAM Roles**

```
AWS Console → IAM → Roles
→ my-live-stream-dev-medialive-role
→ my-live-stream-dev-lambda-role
```

CLI로 확인

```bash
```

```
# S3 Bucket 확인
aws s3 ls | grep my-live-stream

# Security Group 확인
aws ec2 describe-security-groups \
  --filters "Name=group-name,Values=my-live-stream-dev-medialive-input" \
  --query 'SecurityGroups[0].GroupId'

# IAM Role 확인
aws iam list-roles \
  --query 'Roles[?contains(RoleName, `my-live-stream`)].RoleName'
```

📖 코드 세부 학습

Network 모듈 분석

modules/network/main.tf:

```
hcl

# 기본 VPC 조회
data "aws_vpc" "default" {
  count   = var.create_vpc ? 0 : 1
  default = true
}
```

- `count`: 조건부 리소스 생성
- `var.create_vpc ? 0 : 1`: 삼항 연산자
- `default = true`: 기본 VPC만 조회

```
hcl

# Security Group 생성
resource "aws_security_group" "medialive_input" {
  name   = "${var.project_name}-${var.environment}-medialive-input"
  vpc_id = var.create_vpc ? aws_vpc.main[0].id : data.aws_vpc.default[0].id

  ingress {
    from_port   = 1935
    to_port     = 1935
    protocol    = "tcp"
    cidr_blocks = var.allowed_rtmp_cidrs
  }
}
```

- `ingress`: 인바운드 규칙

- `1935`: RTMP 포트
- `cidr_blocks`: 허용할 IP 범위

IAM 모듈 분석

modules/iam/main.tf:

```hcl
resource "aws_iam_role" "medialive" {
  name = "${var.project_name}-${var.environment}-medialive-role"

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

- `assume_role_policy`: 누가 이 Role을 사용할 수 있는지
- `jsonencode()`: JSON 생성
- `Service`: AWS 서비스만 사용 가능

```hcl
```

```
resource "aws_iam_role_policy" "medialive_s3" {
  name = "s3-access"
  role = aws_iam_role.medialive.id

  policy = jsonencode({
    Statement = [{
      Effect = "Allow"
      Action = [
        "s3:PutObject",
        "s3:GetObject"
      ]
      Resource = [
        "${var.archive_bucket_arn}/*"
      ]
    }]
  })
}
```

- `role`: 어떤 Role에 정책 연결
- `Action`: 허용할 작업들
- `Resource`: 어떤 리소스에 대해

Storage 모듈 분석

modules/storage/main.tf:

```hcl
resource "aws_s3_bucket" "archive" {
  bucket = "${var.project_name}-${var.environment}-archive-${data.aws_caller_identity.current.account_id}"
}
```

- S3 Bucket 이름은 전역적으로 고유해야 함
- Account ID 추가로 충돌 방지

```hcl
```

```
resource "aws_s3_bucket_lifecycle_configuration" "archive" {
  bucket = aws_s3_bucket.archive.id

  rule {
    id     = "delete-old-archives"
    status = "Enabled"

    expiration {
      days = var.archive_retention_days
    }
  }
}
```

- Lifecycle Rule: 오래된 파일 자동 삭제
- `days = 7`: 7일 후 삭제
- 비용 절감 효과

💰 Phase 1 비용

생성된 리소스 비용:

- **VPC**: 무료
- **Security Group**: 무료
- **IAM Roles**: 무료
- **S3 Bucket**: $0.023/GB (저장된 데이터만)

예상 비용: 거의 $0 (데이터 저장 전)

🔧 트러블슈팅

문제 1: terraform init 실패

오류:

```
Error: Failed to install provider
```

해결:

```
bash

# 캐시 삭제 후 재시도
rm -rf .terraform .terraform.lock.hcl
terraform init
```

문제 2: AWS 권한 오류

오류:

```
Error: error creating S3 bucket: AccessDenied
```

해결:

```bash
# 권한 확인
aws iam get-user

# 필요한 권한:
# - AmazonS3FullAccess
# - IAMFullAccess
# - AmazonVPCFullAccess
```

문제 3: Bucket 이름 충돌

오류:

```
Error: bucket already exists
```

해결:

```hcl
# terraform.tfvars에서 project_name 변경
project_name = "my-stream-unique-name"
```

문제 4: 리전 오류

오류:

```
Error: operation not supported in region
```

해결:

```hcl
# terraform.tfvars에서 리전 확인
region = "ap-northeast-2"  # 서울 리전
```

🎓 학습 체크리스트

- [ ] Terraform 기본 구조 이해
- [ ] Provider와 Backend 개념 파악
- [ ] 모듈 시스템 이해
- [ ] variables.tf와 terraform.tfvars 차이
- [ ] outputs.tf 역할 이해

- [ ] AWS 리소스 생성 과정 관찰
- [ ] terraform plan/apply 차이 파악
- [ ] State 파일 개념 이해

📚 추가 학습 자료

Terraform 공식 문서:

- Terraform 기초
- AWS Provider
- 모듈 시스템

AWS 리소스 문서:

- VPC 가이드
- IAM 가이드
- S3 가이드

✨ Phase 1 완료!

축하합니다! 첫 번째 Phase를 완료하셨습니다.

달성한 것:

- ✅ Terraform 기본 구조 학습
- ✅ 15개 AWS 리소스 생성
- ✅ 모듈 시스템 이해
- ✅ IaC (Infrastructure as Code) 경험

다음 단계: Phase 3 (MediaPackage)로 진행하거나, 코드를 더 깊이 분석해보세요!

궁금한 점이 있으면 언제든 질문하세요! 🙂