**3차 미니 프로젝트 1조 결과 보고서**

**< 조원 정보 >**

정밝은태양 ( Captain ), 김명진, 남정현, 우희제, 정하늘

**< 프로젝트 개요 >**

**[ 프로젝트명 ]**

**코드와 친해지길 바래**

Terraform, Ansible을 이용한 AWS 아키텍처 설계 및 구축

**[ 주제 ]**

**Code 를 이용한 클라우드 3 Tier Architecture 구축 및 DevOps 서비스 구성**

* IaC를 이용한 3 Tier Infrastructure 구축
* Terraform 이용 인프라 자원 생성 및 관리
* Ansible 이용 서버의 환경 변수 및 상태 관리
* Amazon CloudWatch, Prometheus, Grafana를 활용한 모니터링
* Python의 Flask Framework를 이용한 웹 서비스 구현 및 배포

**[ 선정 이유 ]**

* 각 인프라를 코드로 구축하여 유지보수와 재사용성을 확보
* 인프라를 코드로 정리함으로써 협업과 문서화를 이용하여 지식 공유
* 인프라 구성과 배포를 자동화해 생산성을 높이어 서비스 매우 효율적인 배포
* 로그 모니터링 시스템을 구축 및 분석하여 모니터링에 대한 깊은 이해 도모

**[ 프로젝트 목표 ]**

* IaC를 이용한 클라우드 인프라 구축 & DevOps 서비스 구성

a. Terraform 이용 AWS EC2, VPC, AutoScaling, ELB, RDS, S3 구축

b. Terraform 이용 AWS EKS 구축 (X)

* 로그 관리 시스템 구성

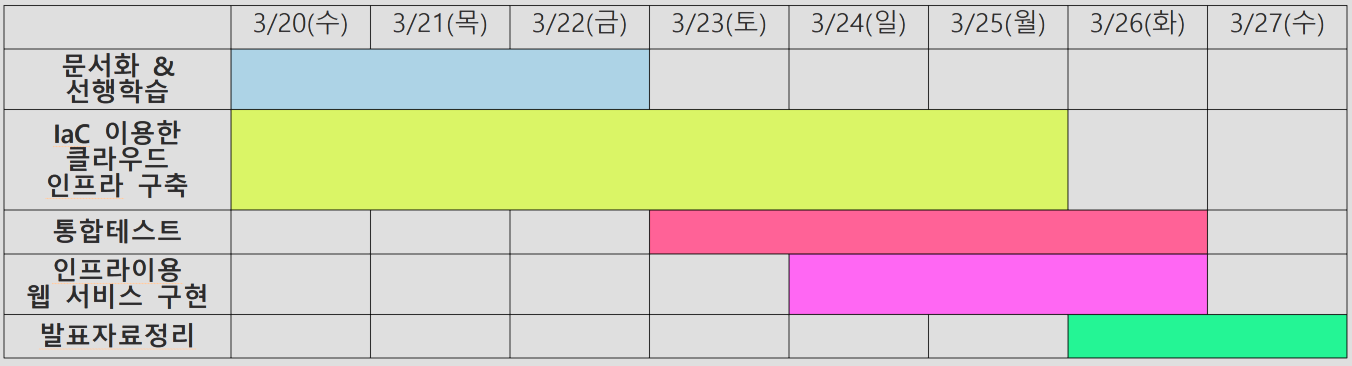
a. Terraform 을 통한 AWS CloudWatch 이용 EC2 서비스 모니터링

b. Ansible을 이용하여 생성된 EC2 인스턴스에 Prometheus, Grafana 모니터링 서비스 구성

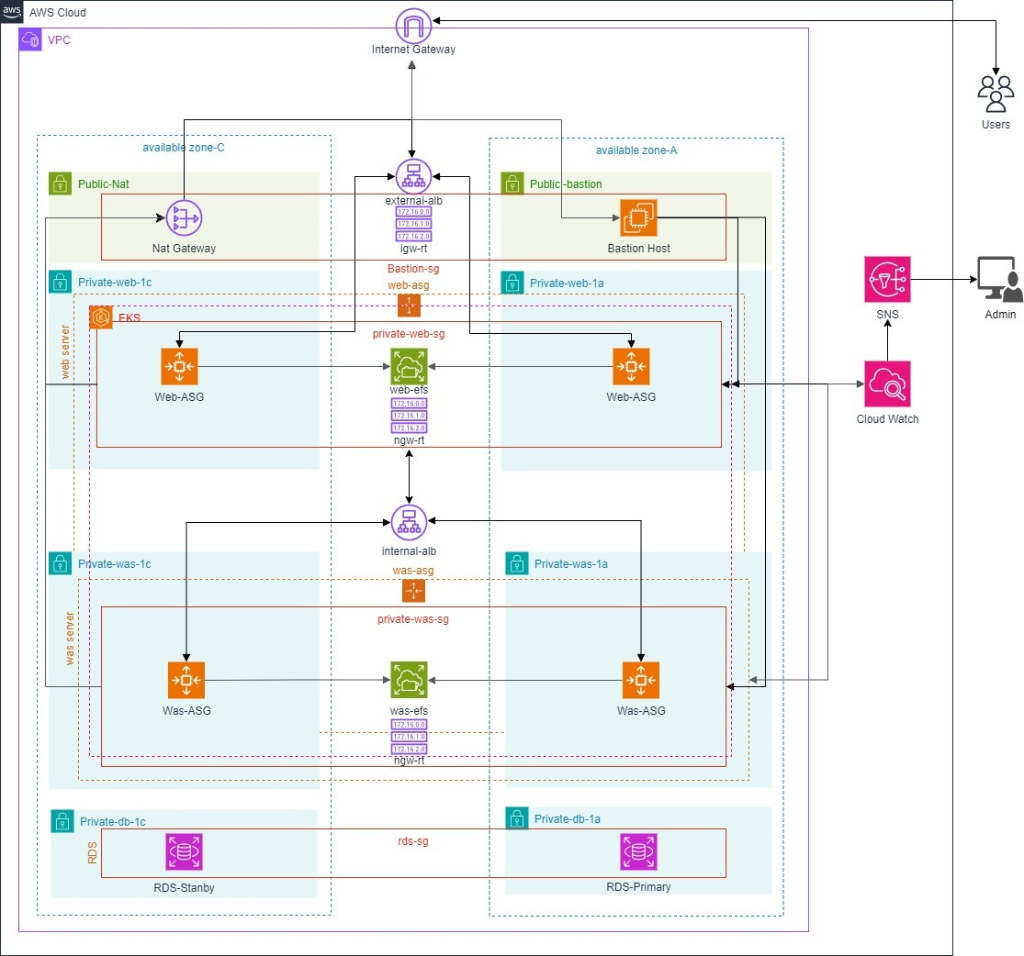
* 인프라 이용 웹 서비스 구현

a. Python의 Flask Framework를 이용한 웹 서비스 구현 및 배포

**[ 프로젝트 일정 ]**



**[아키텍처 구성도]**



**[ 주요 구성 요소 ]**

* Amazon VPC
* Amazon EC2 (Elastic Compute Cloud)
* Amazon ELB
* Amazon RDS
* Amazon EFS
* Amazon ASG
* Cloud Watch

**< 프로젝트 내용 >**

**[ 전체 흐름도 ]**

**||| Terraform for Infra**

**1. AWS VPC를 이용한 가상 네트워크 구축**

Region, VPC, Subnet, Gateway ( IG & NAT ), RoutingTable

**2. AWS Resource를 이용한 서비스 준비**

Security Group, EFS, RDS, Loard Balancer, Key-Pair

**3. Test Bed Intansces 생성 및 서비스 준비**

Bastion, Web, WAS

**4. 앱 서비스 생성 및 실행**

Systemd

**5. Test Bed Intances를 이용해서 이미지 생성**

AMI

**6. 모니터링 및 경보 준비**

Cloud Watch

**7. 이미지와 경보를 이용해서 Auto Scaling 적용**

Auto Scaling Group

**||| Ansible for Monitoring**

**1. 기본 사항**

* Instance 가 생성, 소멸하는 과정에서 시시각각 변하는 ip 에 대응 하기 위해 Dynamic Inventory 활용이 필요
* 각 설치 과정에 맞춰서 Ansible Role을 생성해 관리의 용이하게 할 필요가 있다.
* 각 인스턴스의 상태를 수집하는 Node-Exporter 는 Private Subnet 에 있는 Web / Was Layer 의 Instance 들에 설치 된다.
* Node-Exporter 의 정보를 수집하는 Prometheus 와 Grafana 는 Bastion에 설치해 상대적으로 접근이 쉽게 하였다.

**2. 동적 인벤토리 설정**

* EC2 를 이용하는 Dynamic Inventory 는 끝이 반드시 aws\_ec2.yaml 로 끝나야 함

**3. node-exporter 설치**

**4. Prometheus 설치**

**5. Gratana 설치**

**6. 중요한 부분**

* dynamic inventory

: aws instance 삭제, 재생성, 재부팅 을 하는 경우 public, private ip 가 변하게 된다.기존 static inventory 는 그럴 경우 수동으로 인벤토리를 수정해주어야 하는 단점이 있다.이를 해결하기 위해 dynamic inventory 를 사용해 자동으로 반영되도록 할 수 있다.

* ssh proxy jump

- magic variable

: web / app layer 의 instance 들은 현재 ssh 접근을 bastion에서만 허용하는 security group을 가지고 있다. 그런데 ansible 을 실행하는 곳은 local 이므로 기본적으로 ansible 을 통한 설치가 불가능 하다. 이를 해결 하기 위해 변수로 ssh proxy jump 를 사용하여 bastion 을 거쳐서 ssh 접근을 하도록 설정한다.여기서 다시 bastion 의 public ip는 instance 의 상황에 따라 변할 수 있으므로 동적으로 처리할 수 있어야 한다. 이 때 사용하는 것이 magic variable이다.

**< 코드 및 결과 >**

**||| Terraform for Infra**

**1. 네트워크**

**vpc.tf**

resource "aws\_vpc" "project\_vpc" {

cidr\_block = "10.0.0.0/16"

tags = {

    Name = "project\_VPC"

  }

enable\_dns\_hostnames      = true

enable\_dns\_support        = true

}

**Subnet.tf**

resource "aws\_subnet" "public" {

  count      = length(var.public\_subnet)

  vpc\_id     = aws\_vpc.project\_vpc.id

  cidr\_block = element(var.public\_subnet, count.index)

  availability\_zone = element(var.azs, count.index)

    tags = {

    Name = "project\_Public-Subnet\_0${count.index + 1}"

  }

  map\_public\_ip\_on\_launch = true

}

resource "aws\_subnet" "web" {

  count      = length(var.web\_subnet)

  vpc\_id     = aws\_vpc.project\_vpc.id

  cidr\_block = element(var.web\_subnet, count.index)

  availability\_zone = element(var.azs, count.index)

    tags = {

    Name = "project\_Web-Subnet-0${count.index + 1}"

  }

}

resource "aws\_subnet" "app" {

  count      = length(var.app\_subnet)

  vpc\_id     = aws\_vpc.project\_vpc.id

  cidr\_block = element(var.app\_subnet, count.index)

  availability\_zone = element(var.azs, count.index)

    tags = {

    Name = "project\_App-Subnet-0${count.index + 1}"

  }

}

resource "aws\_subnet" "db" {

  count      = length(var.db\_subnet)

  vpc\_id     = aws\_vpc.project\_vpc.id

  cidr\_block = element(var.db\_subnet, count.index)

  availability\_zone = element(var.azs, count.index)

    tags = {

    Name = "project\_db-Subnet-0${count.index + 1}"

  }

}

**Routingtable.tf**

resource "aws\_route\_table" "public\_rt" {

  vpc\_id = aws\_vpc.project\_vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.igw.id

  }

  tags = {

    Name = "Publinc-Route-Table"

  }

}

resource "aws\_route\_table" "private\_rt" {

  vpc\_id = aws\_vpc.project\_vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    nat\_gateway\_id = aws\_nat\_gateway.ngw.id

  }

  tags = {

    Name = "Private-Route-Table"

  }

}

resource "aws\_route\_table\_association" "public\_subnet\_asso" {

  count = length(var.public\_subnet)

  subnet\_id      = element(aws\_subnet.public[\*].id, count.index)

  route\_table\_id = aws\_route\_table.public\_rt.id

}

resource "aws\_route\_table\_association" "web\_subnet\_asso" {

  count = length(var.web\_subnet)

  subnet\_id      = element(aws\_subnet.web[\*].id, count.index)

  route\_table\_id = aws\_route\_table.private\_rt.id

}

resource "aws\_route\_table\_association" "app\_subnet\_asso" {

  count = length(var.app\_subnet)

  subnet\_id      = element(aws\_subnet.app[\*].id, count.index)

  route\_table\_id = aws\_route\_table.private\_rt.id

}

resource "aws\_route\_table\_association" "db\_subnet\_asso" {

  count = length(var.db\_subnet)

  subnet\_id      = element(aws\_subnet.db[\*].id, count.index)

  route\_table\_id = aws\_route\_table.private\_rt.id

}

**gateway.tf**

resource "aws\_eip" "project\_nat\_eip" {

  vpc = true

  lifecycle {

    create\_before\_destroy = true

  }

}

resource "aws\_internet\_gateway" "igw" {

  vpc\_id = aws\_vpc.project\_vpc.id

  tags = {

    Name = "projectVPC-IGW"

 }

}

resource "aws\_nat\_gateway" "ngw" {

  allocation\_id = aws\_eip.project\_nat\_eip.id

  subnet\_id     = aws\_subnet.public[0].id

  tags = {

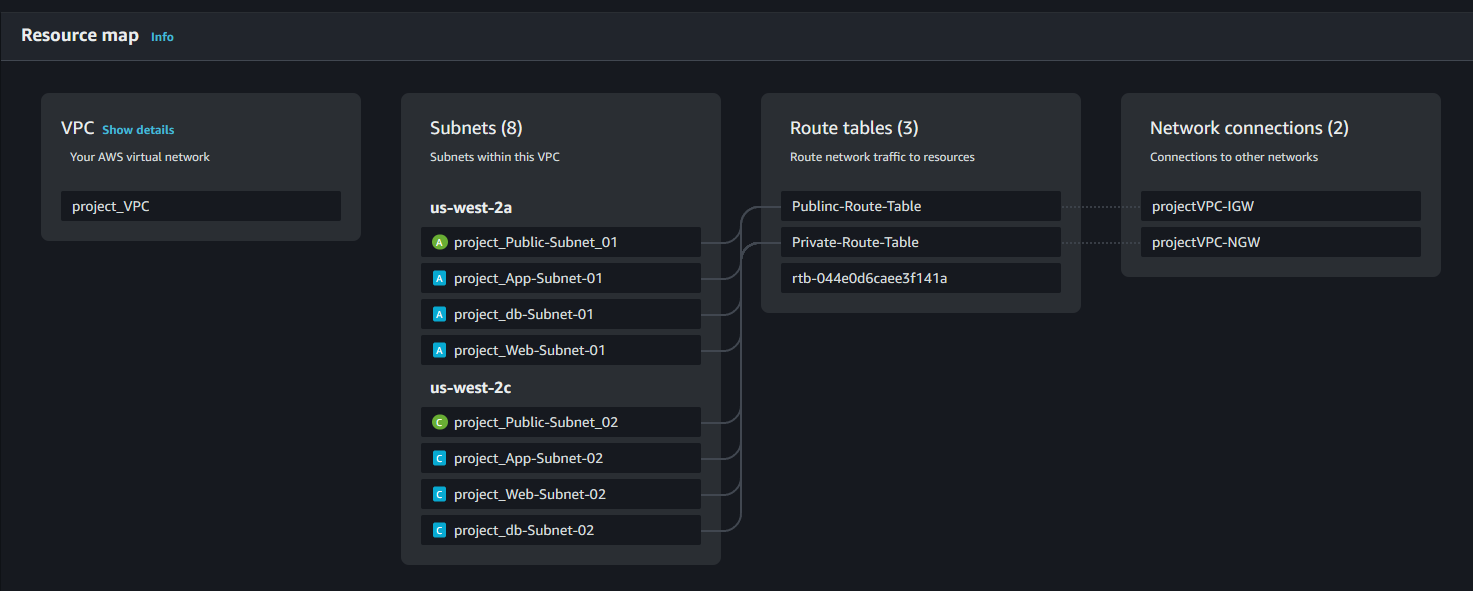
    Name = "projectVPC-NGW"

 }

 depends\_on = [aws\_internet\_gateway.igw]

}

**결과**

****

**2. 인스턴스 ( TestBed & ASG )**

**Instance.tf**

resource "aws\_instance" "project\_bastion" {

  ami = data.aws\_ami.amazon\_linux\_2023.id

  instance\_type = "t2.small"

  vpc\_security\_group\_ids = [aws\_security\_group.project\_bastion.id]

  key\_name = aws\_key\_pair.terraform\_key.key\_name

  subnet\_id = aws\_subnet.public[0].id

  associate\_public\_ip\_address = true

  iam\_instance\_profile = aws\_iam\_instance\_profile.blind\_bastion\_profile.name

  depends\_on = [

    aws\_instance.project\_web,

    aws\_instance.project\_app

    ]

  tags = merge(var.testbed\_tags,var.bastion\_layer\_tags,

    {

      Name = "project\_bastion"

    })

}

resource "aws\_instance" "project\_web" {

  ami = data.aws\_ami.amazon\_linux\_2023.id

  instance\_type = "t2.small"

  vpc\_security\_group\_ids = [aws\_security\_group.project\_web.id]

  key\_name = aws\_key\_pair.terraform\_key.key\_name

  subnet\_id = aws\_subnet.web[0].id

  associate\_public\_ip\_address = false

  iam\_instance\_profile   = aws\_iam\_instance\_profile.testbed\_cloudwatch\_profile.name

  depends\_on=[

    aws\_efs\_file\_system.web\_efs,

    aws\_efs\_mount\_target.web\_mount

    ]

  user\_data = templatefile("./user-data-web.sh",{

    web\_efs\_id = aws\_efs\_file\_system.web\_efs.id

    mount\_point = var.efs\_mount\_point

  })

  tags = merge(var.testbed\_tags,var.web\_layer\_tags,

    {

      Name = "project\_Web"

    }

  )

}

resource "aws\_instance" "project\_app" {

  ami = data.aws\_ami.amazon\_linux\_2023.id

  instance\_type = "t2.small"

  vpc\_security\_group\_ids = [aws\_security\_group.project\_app.id]

  key\_name = aws\_key\_pair.terraform\_key.key\_name

  subnet\_id = aws\_subnet.app[0].id

  associate\_public\_ip\_address = false

  iam\_instance\_profile   = aws\_iam\_instance\_profile.testbed\_cloudwatch\_profile.name

  depends\_on=[

    aws\_efs\_file\_system.app\_efs,

    aws\_efs\_mount\_target.app\_mount

    ]

  user\_data = templatefile("./user-data-app.sh",{

    app\_efs\_id = aws\_efs\_file\_system.app\_efs.id

    mount\_point = var.efs\_mount\_point

  })

  tags = merge(var.testbed\_tags,var.app\_layer\_tags,

    {

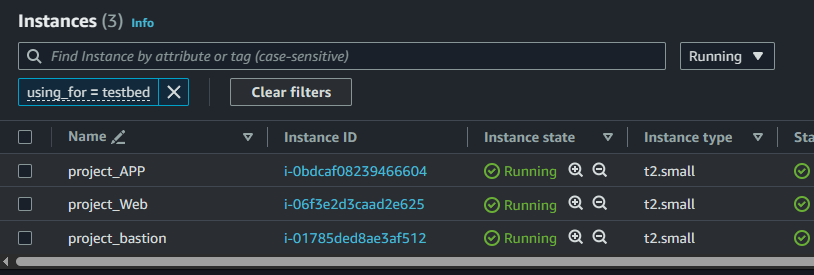
      Name = "project\_APP"

    }

  )

}

**결과**

****

**3. 로드 밸런서**

**lb.tf**

resource "aws\_alb" "external\_lb" {

  name               = "ext-lb"

  load\_balancer\_type = "application"

  internal = false

  subnets = aws\_subnet.public[\*].id

  security\_groups = [aws\_security\_group.project\_ext-lb.id]

}

resource "aws\_lb\_listener" "ext\_listener" {

  load\_balancer\_arn = aws\_alb.external\_lb.arn

  port              = "80"

  protocol          = "HTTP"

  default\_action {

    type             = "forward"

    target\_group\_arn = aws\_lb\_target\_group.ext-tg.arn

  }

}

resource "aws\_lb\_target\_group" "ext-tg" {

  name        = "ext-lb-target-group"

  port        = 5000

  protocol    = "HTTP"

  vpc\_id      = aws\_vpc.project\_vpc.id

  health\_check {

    matcher = "200,301,302"

    path = "/"

    healthy\_threshold = 2

    unhealthy\_threshold = 2

    timeout             = 5   # 5초의 타임아웃

    interval            = 30  # 30초 간격으로 헬스 체크

  }

}

resource "aws\_alb" "internal\_lb" {

  name               = "int-lb"

  internal           = true

  load\_balancer\_type = "application"

  subnets = aws\_subnet.web[\*].id

  security\_groups = [aws\_security\_group.project\_int-lb.id]

}

resource "aws\_lb\_listener" "int\_listener" {

  load\_balancer\_arn = "${aws\_alb.internal\_lb.arn}"

  port              = "5000"

  protocol          = "HTTP"

  default\_action {

    type             = "forward"

    target\_group\_arn = "${aws\_lb\_target\_group.int-tg.arn}"

  }

}

resource "aws\_lb\_target\_group" "int-tg" {

  name        = "int-lb-target-group"

  port        = 5000

  protocol    = "HTTP"

  # target\_type = "ip"

  vpc\_id      = aws\_vpc.project\_vpc.id

  health\_check {

    matcher = "200,301,302"

    path = "/"

    healthy\_threshold = 2

    unhealthy\_threshold = 2

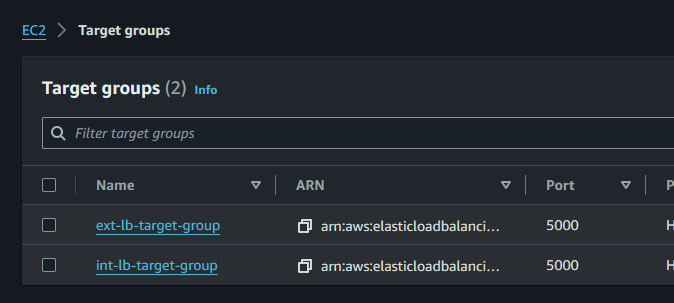
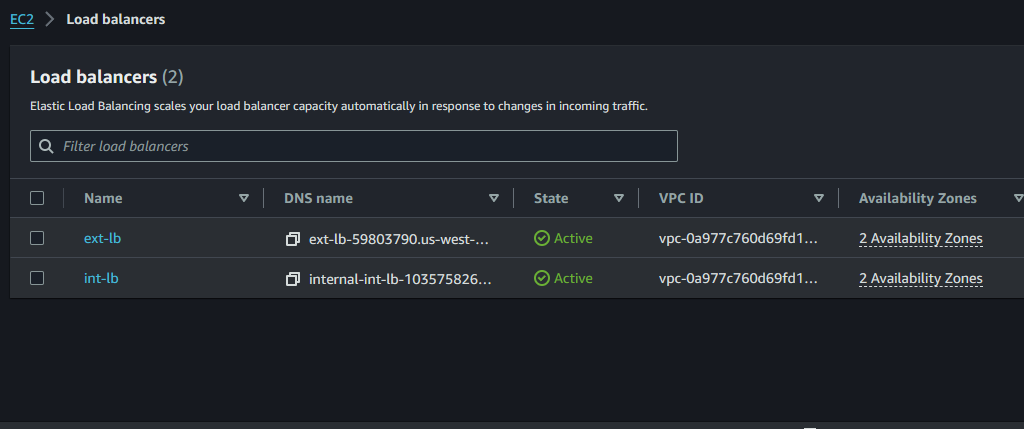
    timeout             = 5   # 5초의 타임아웃

    interval            = 30  # 30초 간격으로 헬스 체크

  }

}

**결과**

****

**4. EFS**

**efs.tf**

resource "aws\_efs\_file\_system" "web\_efs" {

  creation\_token = "web\_efs"

  encrypted = true

  performance\_mode = "generalPurpose"

  throughput\_mode = "bursting"

  lifecycle\_policy {

    transition\_to\_ia = "AFTER\_30\_DAYS"

  }

  tags = {

    Name = "Web\_EFS"

  }

}

resource "aws\_efs\_file\_system" "app\_efs" {

  creation\_token = "app\_efs"

  encrypted = true

  performance\_mode = "generalPurpose"

  throughput\_mode = "bursting"

  lifecycle\_policy {

    transition\_to\_ia = "AFTER\_30\_DAYS"

  }

  tags = {

    Name = "APP\_EFS"

  }

}

resource "aws\_efs\_mount\_target" "web\_mount" {

    count = length(var.web\_subnet)

    file\_system\_id  = aws\_efs\_file\_system.web\_efs.id

    subnet\_id       = element(aws\_subnet.web[\*].id, count.index)

    security\_groups = [aws\_security\_group.project\_web\_efs.id]

}

resource "aws\_efs\_mount\_target" "app\_mount" {

    count = length(var.app\_subnet)

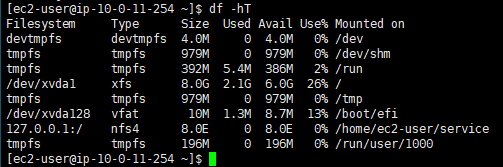
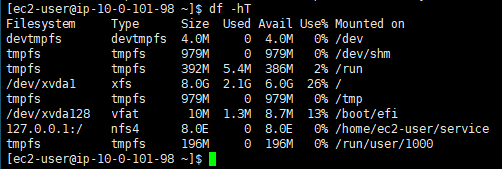
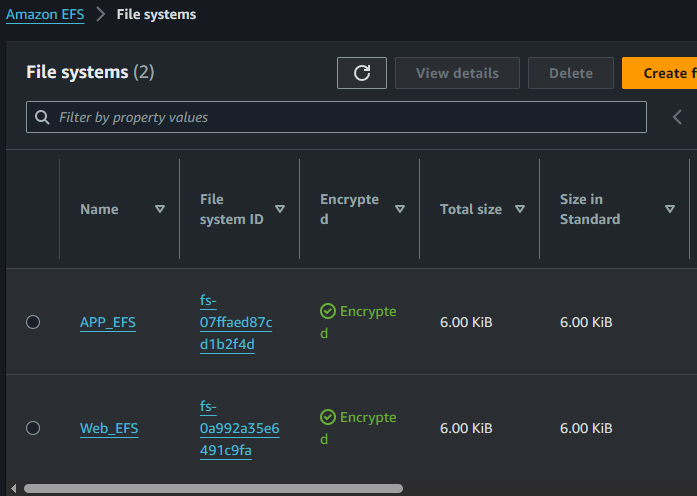
    file\_system\_id  = aws\_efs\_file\_system.app\_efs.id

    subnet\_id      = element(aws\_subnet.app[\*].id, count.index)

    security\_groups = [aws\_security\_group.project\_app\_efs.id]

}

**결과**

****

**5. RDS**

**rds.tf**

resource "aws\_db\_instance" "blind\_rds" {

  identifier = "project-blind-db"

  allocated\_storage = 50

  max\_allocated\_storage = 100

  engine = "mysql"

  engine\_version = "8.0.35"

  instance\_class = "db.t3.micro"

  db\_name  = "blind" # Initial database name

  username = "${var.db\_user\_name}"

  password = "${var.db\_user\_pass}"

  multi\_az = true

  publicly\_accessible = false

  skip\_final\_snapshot = true

  db\_subnet\_group\_name        = aws\_db\_subnet\_group.default.id

  vpc\_security\_group\_ids = [ aws\_security\_group.project\_db.id ]

  tags = {

      Name = "Blind DB"

  }

}

resource "aws\_db\_subnet\_group" "default" {

  name       = "db-subnet-group"

  subnet\_ids = aws\_subnet.db[\*].id

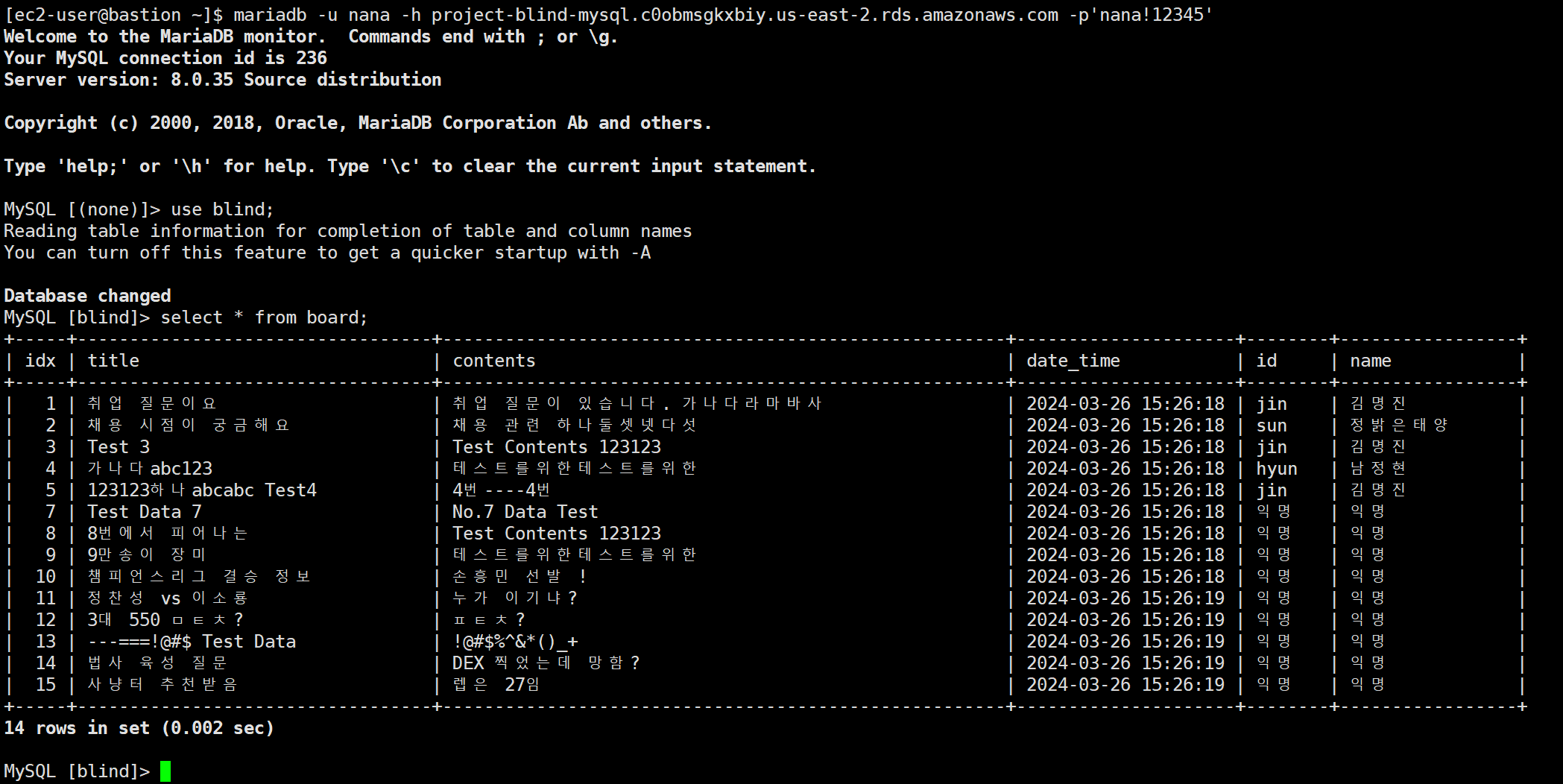
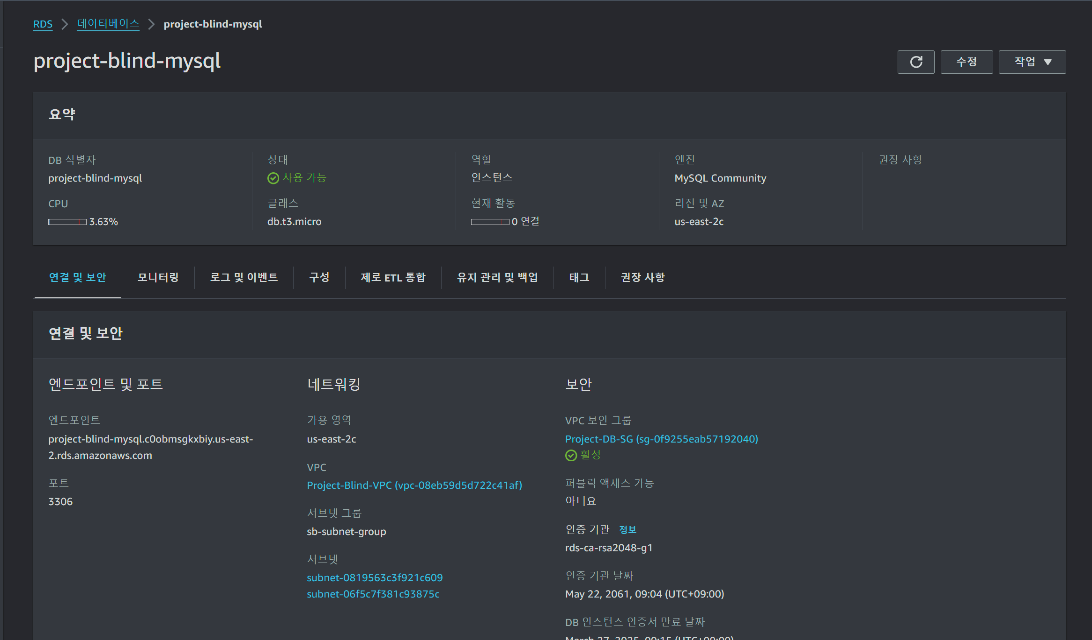
  tags = {

    Name = "My DB subnet group"

  }

}

**결과**



**6. 서비스**

**service.tf**

resource "null\_resource" "copy\_files" {

  connection {

    type = "ssh"

    user = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "file" {

    source = var.private\_key\_location

    destination = "${var.dest1}"

  }

  provisioner "file" {

    source = "./service"

    destination = "${var.dest2}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo chmod 400 ${var.dest1}"

    ]

  }

  depends\_on = [aws\_instance.project\_bastion]

}

resource "null\_resource" "define\_lb\_dns" {

  connection {

    type        = "ssh"

    user        = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host        = "${aws\_instance.project\_bastion.public\_ip}"

  }

  count = length(var.flask\_file\_paths)

  provisioner "remote-exec" {

    inline = [

      "sed -i 's/INT\_LB\_DNS/${aws\_alb.internal\_lb.dns\_name}/g'  ${element(var.flask\_file\_paths, count.index)}"

    ]

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files,

    aws\_alb.internal\_lb,

  ]

}

resource "null\_resource" "define\_rds\_end\_point" {

  connection {

    type        = "ssh"

    user        = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host        = "${aws\_instance.project\_bastion.public\_ip}"

  }

  count = length(var.dao\_file\_paths)

  provisioner "remote-exec" {

    inline = [

      "sed -i 's/RDS\_END\_POINT/${aws\_db\_instance.blind\_rds.address}/g' ${element(var.dao\_file\_paths, count.index)}"

    ]

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files,

    aws\_db\_instance.blind\_rds,

  ]

}

resource "null\_resource" "web\_service\_copy" {

  connection {

    type = "ssh"

    user = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo chown ec2-user:ec2-user -R ${var.efs\_mount\_point}'",

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/blind\_web ec2-user@${aws\_instance.project\_web.private\_ip}:${var.efs\_mount\_point}",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo chmod 755 -R ${var.efs\_mount\_point}'",

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/web.tpl ec2-user@${aws\_instance.project\_web.private\_ip}:${var.efs\_mount\_point}/blind\_web.service",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo cp ${var.efs\_mount\_point}/blind\_web.service /etc/systemd/system/blind\_web.service'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo systemctl daemon-reload'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo chmod 755 ${var.efs\_mount\_point}/blind\_web.service'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo systemctl enable --now blind\_web.service'"

    ]

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files,

    null\_resource.define\_lb\_dns,

    null\_resource.define\_rds\_end\_point

    ]

}

resource "null\_resource" "app\_service\_copy" {

  connection {

    type = "ssh"

    user = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo chown ec2-user:ec2-user -R ${var.efs\_mount\_point}'",

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/blind\_was ec2-user@${aws\_instance.project\_app.private\_ip}:${var.efs\_mount\_point}",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo chmod 755 -R ${var.efs\_mount\_point}'",

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/was.tpl ec2-user@${aws\_instance.project\_app.private\_ip}:${var.efs\_mount\_point}/blind\_was.service",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo chmod 755 ${var.efs\_mount\_point}/blind\_was.service'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo cp ${var.efs\_mount\_point}/blind\_was.service /etc/systemd/system/blind\_was.service'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo systemctl daemon-reload'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo systemctl enable --now blind\_was.service'"

    ]

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files,

    null\_resource.define\_rds\_end\_point

    ]

}

resource "null\_resource" "input\_dummy\_data" {

  connection {

    type        = "ssh"

    user        = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host        = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/dummy\_data.sql ec2-user@${aws\_instance.project\_app.private\_ip}:${var.dest\_dummy\_data}",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo mariadb -u ${var.db\_user\_name} -p'${var.db\_user\_pass}' -P 3306 -h ${aws\_db\_instance.blind\_rds.address} blind < ${var.dest\_dummy\_data}'"

    ]

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files,

    aws\_db\_instance.blind\_rds,

    null\_resource.app\_service\_copy

  ]

}

resource "null\_resource" "delete\_service\_bastion" {

  connection {

    type        = "ssh"

    user        = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host        = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo rm -rf ${var.dest2}"

    ]

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files,

    null\_resource.web\_service\_copy,

    null\_resource.app\_service\_copy,

    null\_resource.input\_dummy\_data,

    null\_resource.install\_cloudwatch\_web,

    null\_resource.install\_cloudwatch\_app

  ]

}

resource "null\_resource" "install\_monitoring" {

  provisioner "local-exec" {

    working\_dir = "./ansible"

    command = "ANSIBLE\_HOST\_KEY\_CHECKING=False ansible-playbook playbook.yaml"

  }

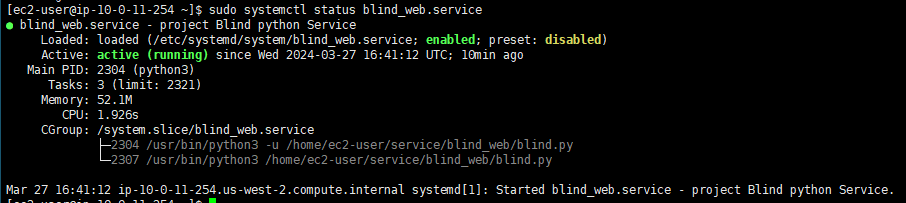
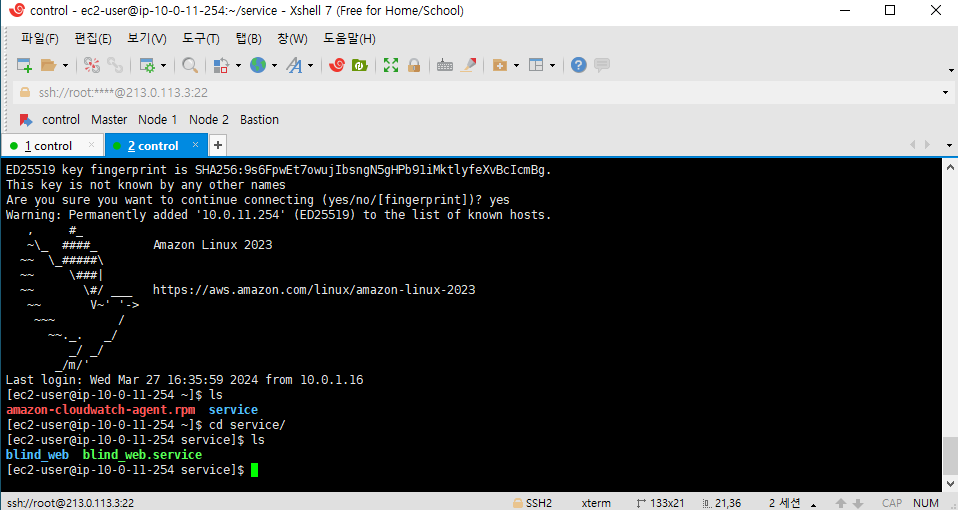
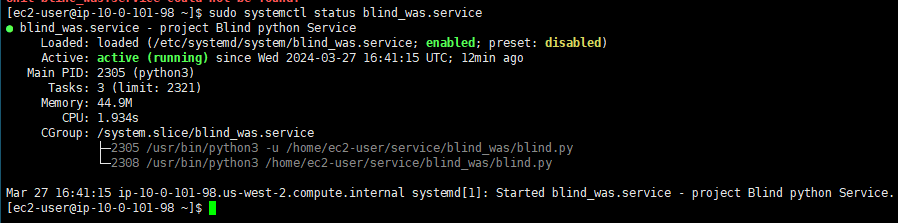
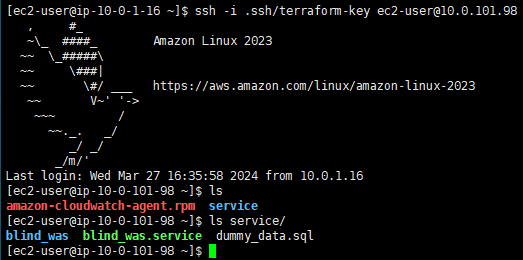
  depends\_on = [

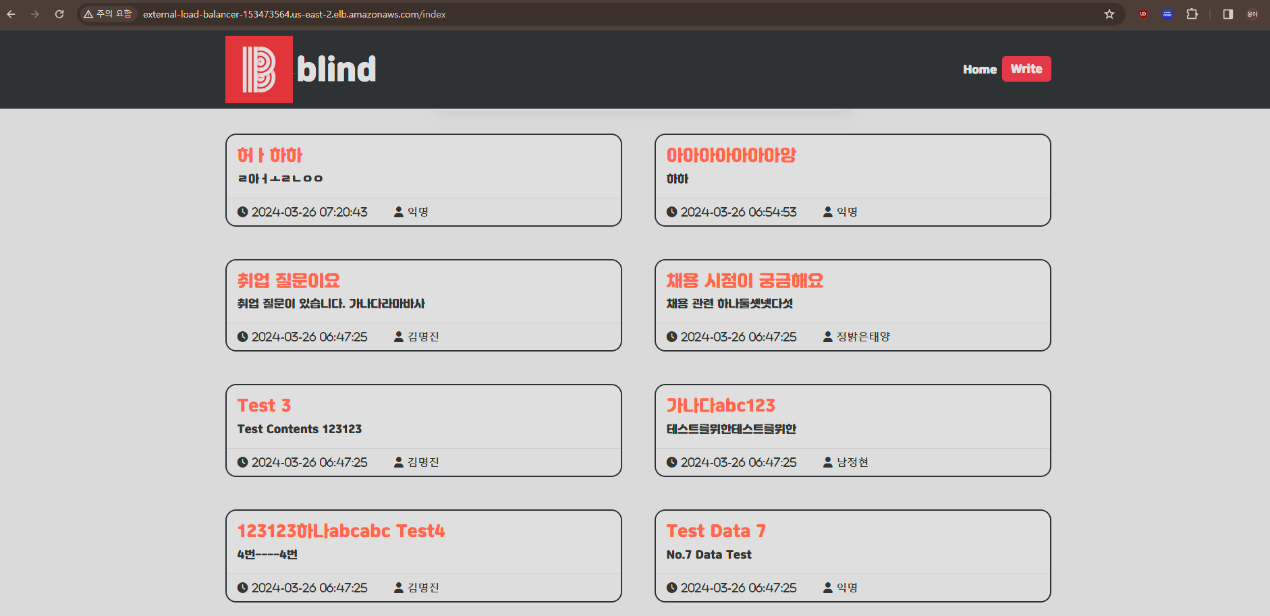
      null\_resource.delete\_service\_bastion

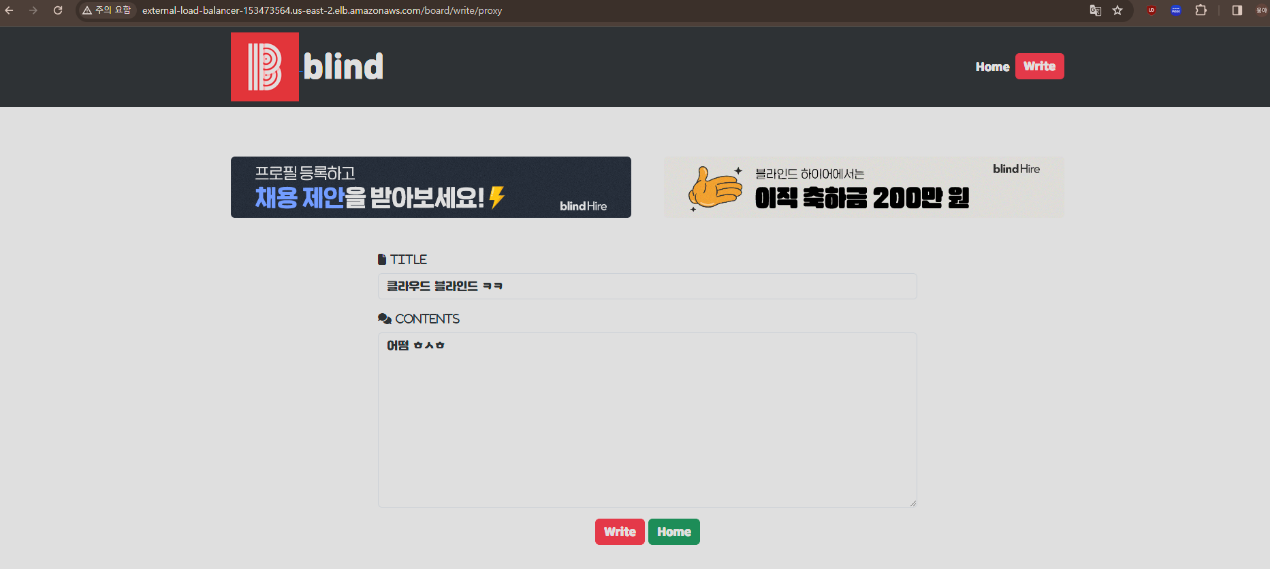
    ]

}

**결과**

****





**7. AMI**

**make\_ami.tf**

resource "aws\_iam\_role" "blind\_bastion\_role" {

  name               = "sk104-003-bastion-cloudwatch-terraform"

  assume\_role\_policy = <<EOF

{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Effect": "Allow",

      "Principal": {

        "Service": "ec2.amazonaws.com"

      },

      "Action": "sts:AssumeRole"

     }

   ]

  }

  EOF

}

resource "aws\_iam\_role" "testbed\_cloudwatch\_role" {

  name               = "sk104-003-testbed-cloudwatch-terraform"

  assume\_role\_policy = <<EOF

{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Effect": "Allow",

      "Principal": {

        "Service": "ec2.amazonaws.com"

      },

      "Action": "sts:AssumeRole"

     }

   ]

  }

  EOF

}

resource "aws\_iam\_policy" "bind\_ec2\_policy" {

  name        = "blind\_ec2\_policy"

  path        = "/"

  description = "My test policy"

  policy = jsonencode({

    Version = "2012-10-17"

    Statement = [

      {

        Action = [

          "ec2:Describe\*"

        ]

        Effect   = "Allow"

        Resource = "\*"

      },

    ]

  })

}

resource "aws\_iam\_policy\_attachment" "blind\_bastion\_ec2\_policy" {

  name       = "AmazonEC2RoleforSSM-Attachment"

  roles      = [aws\_iam\_role.blind\_bastion\_role.name]

  policy\_arn = aws\_iam\_policy.bind\_ec2\_policy.arn

}

resource "aws\_iam\_policy\_attachment" "attach\_amazonec2\_policy" {

  name       = "AmazonEC2RoleforSSM-Attachment"

  roles      = [aws\_iam\_role.testbed\_cloudwatch\_role.name]

  policy\_arn = "arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM"

}

resource "aws\_iam\_policy\_attachment" "attach\_cloudwatch\_policy" {

  name       = "CloudWatchAgentServerPolicy-Attachment"

  roles      = [aws\_iam\_role.testbed\_cloudwatch\_role.name]

  policy\_arn = "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"

}

resource "aws\_iam\_instance\_profile" "blind\_bastion\_profile" {

  name = "blind\_bastion-profile"

  role = aws\_iam\_role.blind\_bastion\_role.name

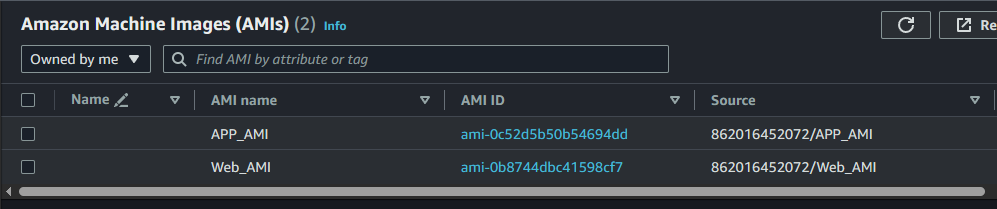
}

resource "aws\_iam\_instance\_profile" "testbed\_cloudwatch\_profile" {

  name = "testbed-cloudwatch-profile"

  role = aws\_iam\_role.testbed\_cloudwatch\_role.name

}

**결과**

**8. Auto Scaling Group**

**launch\_conf.tf**

resource "aws\_launch\_configuration" "web" {

  name\_prefix = "web-"

  image\_id = aws\_ami\_from\_instance.ami\_web.id

  instance\_type = "t2.small"

  key\_name = aws\_key\_pair.terraform\_key.key\_name

  security\_groups = [aws\_security\_group.project\_web.id]

  associate\_public\_ip\_address = false

  root\_block\_device {

    volume\_type = "standard"

    volume\_size = 12

  }

  ebs\_block\_device {

    device\_name = "/dev/sdb"

    volume\_type = "standard"

    volume\_size = 10

    encrypted   ="false"

  }

  lifecycle {

    create\_before\_destroy = true

  }

}

resource "aws\_launch\_configuration" "app" {

  name\_prefix = "app-"

  image\_id = aws\_ami\_from\_instance.ami\_app.id

  instance\_type = "t2.small"

  key\_name = aws\_key\_pair.terraform\_key.key\_name

  security\_groups = [aws\_security\_group.project\_app.id]

  associate\_public\_ip\_address = false

  root\_block\_device {

    volume\_type = "standard"

    volume\_size = 12

  }

  ebs\_block\_device {

    device\_name = "/dev/sdb"

    volume\_type = "standard"

    volume\_size = 10

    encrypted   ="false"

  }

  # user\_data = templatefile("./user-data-app.sh",{

  #   app\_efs\_id = aws\_efs\_file\_system.app\_efs.id

  #   mount\_point = var.efs\_mount\_point

  # })

  lifecycle {

    create\_before\_destroy = true

  }

}

**asg.tf**

resource "aws\_autoscaling\_group" "web" {

  name = "${aws\_launch\_configuration.web.name}-asg"

  min\_size             = 1

  desired\_capacity     = 2

  max\_size             = 4

  target\_group\_arns = [ aws\_lb\_target\_group.ext-tg.arn ]

  health\_check\_type    = "EC2"

  launch\_configuration = aws\_launch\_configuration.web.name

  enabled\_metrics = [

    "GroupMinSize",

    "GroupMaxSize",

    "GroupDesiredCapacity",

    "GroupInServiceInstances",

    "GroupTotalInstances"

  ]

  metrics\_granularity = "1Minute"

  vpc\_zone\_identifier  = aws\_subnet.web[\*].id

  lifecycle {

    create\_before\_destroy = true

  }

  tag {

      key = "Name"

      value = "Web"

      propagate\_at\_launch = true

    }

  dynamic "tag" {

    for\_each = var.web\_asg\_tags

    content {

      key    =  tag.key

      value   =  tag.value

      propagate\_at\_launch =  true

    }

  }

  depends\_on = [

    aws\_alb.external\_lb,

    aws\_ami\_from\_instance.ami\_web

   ]

}

resource "aws\_autoscaling\_group" "app" {

  name = "${aws\_launch\_configuration.app.name}-asg"

  min\_size             = 1

  desired\_capacity     = 2

  max\_size             = 4

  target\_group\_arns = [ aws\_lb\_target\_group.int-tg.arn ]

  health\_check\_type    = "EC2"

  launch\_configuration = aws\_launch\_configuration.app.name

  enabled\_metrics = [

    "GroupMinSize",

    "GroupMaxSize",

    "GroupDesiredCapacity",

    "GroupInServiceInstances",

    "GroupTotalInstances"

  ]

  metrics\_granularity = "1Minute"

  vpc\_zone\_identifier  = aws\_subnet.app[\*].id

  lifecycle {

    create\_before\_destroy = true

  }

  tag {

      key ="Name"

      value = "APP"

      propagate\_at\_launch = true

    }

  dynamic "tag" {

    for\_each = var.app\_asg\_tags

    content {

      key    =  tag.key

      value   =  tag.value

      propagate\_at\_launch =  true

    }

  }

  depends\_on = [

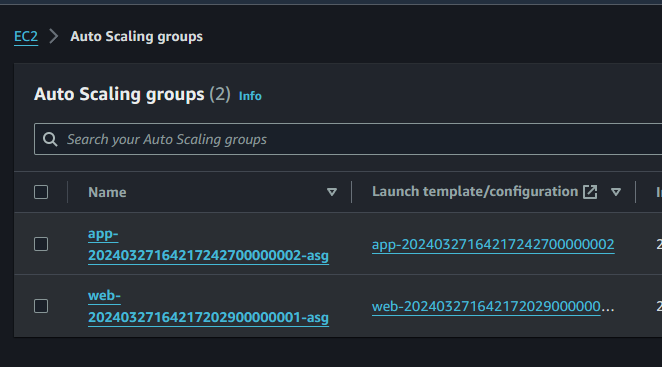
    aws\_alb.internal\_lb,

    aws\_ami\_from\_instance.ami\_app

  ]

}

**결과**

****

**9. Cloud Watch**

**cloud\_watch.tf**

resource "aws\_cloudwatch\_log\_group" "web\_blind\_access" {

  name              = "web\_blind\_access" # The name of the log group.

  retention\_in\_days = 14 # Retention of logs in days.

}

resource "aws\_cloudwatch\_log\_group" "web\_blind\_error" {

  name              = "web\_blind\_error" # The name of the log group.

  retention\_in\_days = 14 # Retention of logs in days.

}

resource "aws\_cloudwatch\_log\_group" "app\_blind\_access" {

  name              = "app\_blind\_access" # The name of the log group.

  retention\_in\_days = 14 # Retention of logs in days.

}

resource "aws\_cloudwatch\_log\_group" "app\_blind\_error" {

  name              = "app\_blind\_error" # The name of the log group.

  retention\_in\_days = 14 # Retention of logs in days.

}

resource "aws\_autoscaling\_policy" "web\_scale\_out" {

  name                   = "scale\_out\_policy"

  scaling\_adjustment     = 1

  adjustment\_type        = "ChangeInCapacity"

  cooldown               = 300

  autoscaling\_group\_name = aws\_autoscaling\_group.web.name

}

resource "aws\_autoscaling\_policy" "app\_scale\_out" {

  name                   = "scale\_out\_policy"

  scaling\_adjustment     = 1

  adjustment\_type        = "ChangeInCapacity"

  cooldown               = 300

  autoscaling\_group\_name = aws\_autoscaling\_group.app.name

}

resource "aws\_cloudwatch\_metric\_alarm" "cpu\_alarm-web" {

  alarm\_name          = "CPUUtilizationHigh\_Web"

  comparison\_operator = "GreaterThanOrEqualToThreshold"

  evaluation\_periods  = "1"

  metric\_name         = "CPUUtilization"

  namespace           = "AWS/EC2"

  period              = "60"

  statistic           = "Average"

  threshold           = "70"

  alarm\_description   = "This metric monitors CPU utilization"

  alarm\_actions       = [aws\_autoscaling\_policy.web\_scale\_out.arn]

}

resource "aws\_cloudwatch\_metric\_alarm" "cpu\_alarm-app" {

  alarm\_name          = "CPUUtilizationHigh\_APP"

  comparison\_operator = "GreaterThanOrEqualToThreshold"

  evaluation\_periods  = "1"

  metric\_name         = "CPUUtilization"

  namespace           = "AWS/EC2"

  period              = "60"

  statistic           = "Average"

  threshold           = "70"

  alarm\_description   = "This metric monitors CPU utilization"

  alarm\_actions       = [aws\_autoscaling\_policy.web\_scale\_out.arn]

}

resource "aws\_sns\_topic" "example\_topic\_web" {

  name = "example-topic\_web"

}

resource "aws\_sns\_topic" "example\_topic\_app" {

  name = "example-topic\_app"

}

resource "aws\_sns\_topic\_subscription" "email\_subscription\_web" {

  topic\_arn = aws\_sns\_topic.example\_topic\_web.arn

  protocol  = "email"

  endpoint  = "illidan2000@naver.com" # 여기에 이메일 주소를 입력하세요

}

resource "aws\_sns\_topic\_subscription" "email\_subscription\_app" {

  topic\_arn = aws\_sns\_topic.example\_topic\_app.arn

  protocol  = "email"

  endpoint  = "illidan2000@naver.com" # 여기에 이메일 주소를 입력하세요

}

resource "null\_resource" "copy\_config\_file" {

  connection {

    type = "ssh"

    user = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "file" {

    source = "./config.json\_web.j2"

    destination = "${var.dest2}/"

  }

  provisioner "file" {

    source = "./config.json\_app.j2"

    destination = "${var.dest2}/"

  }

  depends\_on = [

    aws\_instance.project\_bastion,

    null\_resource.copy\_files

  ]

}

resource "null\_resource" "install\_cloudwatch\_web" {

  connection {

    type = "ssh"

    user = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo yum install -y amazon-cloudwatch-agent'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo systemctl enable --now amazon-cloudwatch-agent'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo wget https://s3.amazonaws.com/amazoncloudwatch-agent/amazon\_linux/amd64/latest/amazon-cloudwatch-agent.rpm'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo rpm -U ./amazon-cloudwatch-agent.rpm'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo chown ec2-user /opt/aws/amazon-cloudwatch-agent/bin'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo yum install collectd -y'",

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/config.json\_web.j2 ec2-user@${aws\_instance.project\_web.private\_ip}:/opt/aws/amazon-cloudwatch-agent/bin/config.json",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo chown ec2-user:ec2-user /opt/aws/amazon-cloudwatch-agent/bin/config.json'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo chmod 755 /opt/aws/amazon-cloudwatch-agent/bin/config.json'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json -s'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo systemctl restart amazon-cloudwatch-agent'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_web.private\_ip} 'sudo systemctl enable --now collectd'"

    ]

  }

  depends\_on = [ null\_resource.copy\_config\_file ]

}

resource "null\_resource" "install\_cloudwatch\_app" {

  connection {

    type = "ssh"

    user = "ec2-user"

    private\_key = "${file("~/.ssh/terraform-key")}"

    host = "${aws\_instance.project\_bastion.public\_ip}"

  }

  provisioner "remote-exec" {

    inline = [

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo yum install -y amazon-cloudwatch-agent'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo systemctl enable --now amazon-cloudwatch-agent'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo wget https://s3.amazonaws.com/amazoncloudwatch-agent/amazon\_linux/amd64/latest/amazon-cloudwatch-agent.rpm'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo rpm -U ./amazon-cloudwatch-agent.rpm'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo chown ec2-user /opt/aws/amazon-cloudwatch-agent/bin'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo yum install collectd -y'",

      "sudo scp -i ${var.dest1} -o StrictHostKeyChecking=no -r ${var.dest2}/config.json\_app.j2 ec2-user@${aws\_instance.project\_app.private\_ip}:/opt/aws/amazon-cloudwatch-agent/bin/config.json",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo chown ec2-user:ec2-user /opt/aws/amazon-cloudwatch-agent/bin/config.json'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo chmod 755 /opt/aws/amazon-cloudwatch-agent/bin/config.json'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json -s'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo systemctl restart amazon-cloudwatch-agent'",

      "sudo ssh -i ${var.dest1} -o StrictHostKeyChecking=no ec2-user@${aws\_instance.project\_app.private\_ip} 'sudo systemctl enable --now collectd'"

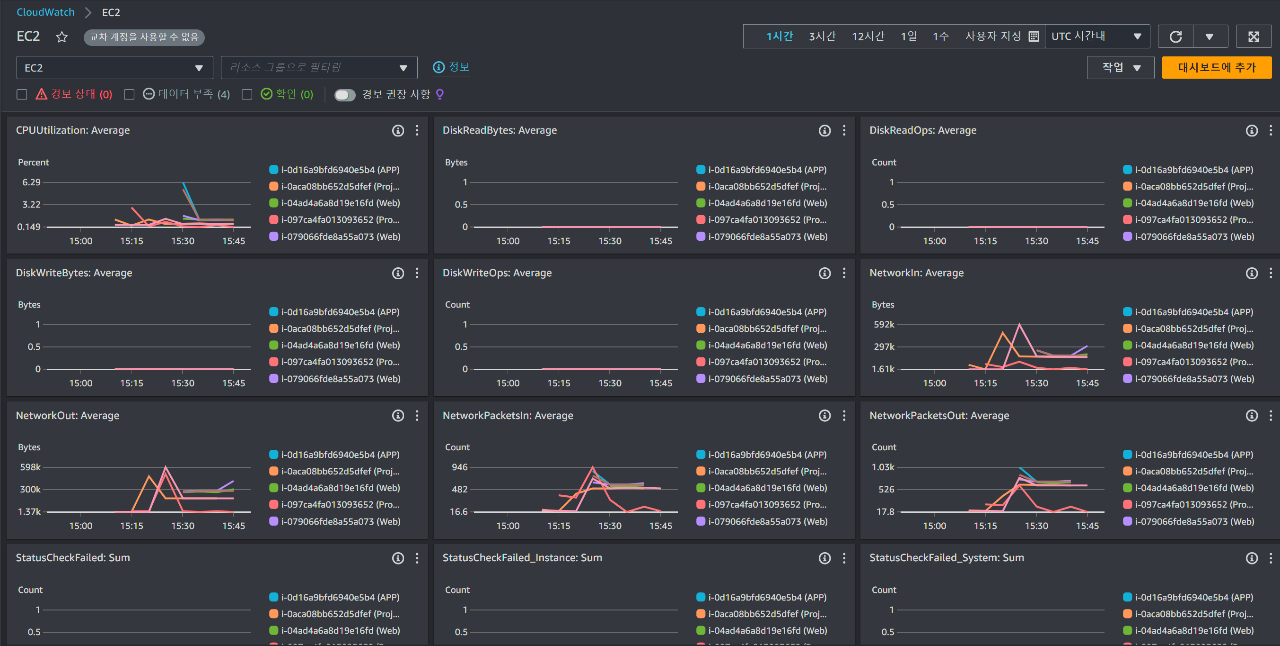
    ]

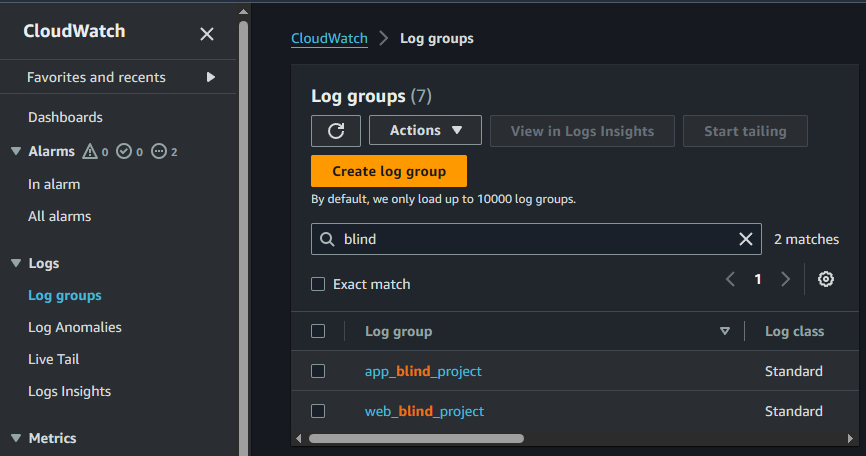
  }

  depends\_on = [ null\_resource.copy\_config\_file ]

}

**결과**



****

**||| Ansible for Monitoring**

**[ Ansible Default Files ]**

**ansible.cfg**

[defaults]

inventory = ~/project\_3rd/ansible/aws\_ec2.yaml

remote\_user = ec2-user

pipelineing = True

ask\_pass = false

command\_warnings=false

private\_key\_file = ~/.ssh/terraform-key

[privilege\_escalation]

become = true

become\_method = sudo

become\_user = root

become\_ask\_pass = false

[inventory]

enable\_plugins = host\_list, script, auto, yaml, ini, toml, aws\_ec2

**aws\_ec2.yaml**

plugin: aws\_ec2

regions:

  - us-west-2

cache: true

cache\_max\_age: 3600

filters:

  tag:Group:

    - bastion

    - private

  instance-state-name: running

keyed\_groups:

  - prefix: tags

    key: tags

hostnames:

  - network-interface.association.public-ip

  - network-interface.addresses.private-ip-address

**cred.yaml**

access\_key: " #Add Access\_Key "

secret\_key: " #Add Secret\_Key "

**playbook.yaml**

---

- name: install node\_exporter on private

  hosts: tags\_Group\_private

  vars:

    ansible\_ssh\_common\_args:

      -o ProxyCommand="ssh

      -o StrictHostKeyChecking=no

      -o UserKnownHostsFile=/dev/null

      -i ~/.ssh/terraform-key

      -W %h:%p ec2-user@{{ hostvars[groups['tags\_Group\_bastion'][0]]["public\_ip\_address"] }}"

  become: True

  gather\_facts: yes

  roles:

    - role: role/node\_exporter

- name: install prometheus & grafana on bastion

  hosts: tags\_Group\_bastion

  become: True

  gather\_facts: yes

  roles:

    - role/prometheus

    - role/grafana

**[ Roles ]**

**Grafana**

**tasks/main.yml**

---

# tasks file for grafana

- name: Create user for Grafana

  user:

    name: "{{ grafana\_user }}"  # Username

    state: present  # Ensure the user is present

    createhome: no  # Create the user's home directory

    shell: /usr/sbin/nologin

    append: yes

  become: true

- name: Copying the repo

  template:

    src: templates/grafana.repo.j2

    dest: "{{ repo\_dest\_dir }}/grafana.repo"

    owner: "root"

    group: "root"

    mode: '0644'

# - name: dnf update

#  ansible.builtin.shell: yum update

#  become: true

- name: Install Grafana

  ansible.builtin.yum:

    name: grafana

    state: present

- name: Daemon reload

  ansible.builtin.systemd\_service:

    daemon\_reload: true

- name: Start the Grafana

  ansible.builtin.systemd\_service:

    name: grafana-server

    state: started

    enabled: yes

#- name: Copying the scrape config

#  template:

#    src: templates/prometheus.yml.j2

#    dest: "{{ prometheus\_dest\_dir }}/prometheus.yml"

#    owner: "{{ grafana\_user }}"

#    group: "{{ grafana\_user }}"

#    mode: '0775'

- name: "start and enable service"

#   when: copy\_conf\_status.changed == true or copy\_status.changed == true

  ansible.builtin.systemd\_service:

    name: grafana-server

    state: restarted

    enabled: yes

    daemon\_reload: yes

**templates/** **grafana.repo.j2**

[grafana]

name=grafana

baseurl=https://packages.grafana.com/oss/rpm

repo\_gpgcheck=1

enabled=1

gpgcheck=1

gpgkey=https://packages.grafana.com/gpg.key

sslverify=1

sslcacert=/etc/pki/tls/certs/ca-bundle.crt

**vars/main.yml**

---

# vars file for grafana

grafana\_user: grafana

repo\_dest\_dir: /etc/yum.repos.d/

**Node Exporter**

**tasks/main.yml**

- name: Create user for node-exporter

  user:

    name: "{{ node\_exporter\_user }}"  # Username

    state: present  # Ensure the user is present

    createhome: no  # Create the user's home directory

    shell: /usr/sbin/nologin

    append: yes

  become: true

- name: Download/unarchive Packages for node-exporter

  unarchive:

    src: "{{ node\_exporter\_download\_url }}"

    dest: /tmp

    owner: "{{ node\_exporter\_user }}"

    group: "{{ node\_exporter\_user }}"

    remote\_src: yes

    creates: "{{ node\_exporter\_download\_dir }}"

- name: Copying the service binary for node-exporter

  copy:

    src: "{{ node\_exporter\_download\_dir }}/{{ item }}"

    dest: /usr/local/bin/

    owner: "{{ node\_exporter\_user }}"

    group: "{{ node\_exporter\_user }}"

    mode: '0775'

    remote\_src: yes

  with\_items:

    - node\_exporter

- name: Removing the tar file of node-exporter

  file:

    path: "{{ node\_exporter\_download\_dir }}"

    state: absent

- name: Create prometheus systemd service file

  template:

    src: templates/node-exporter.service.j2

    dest: "{{ service\_dest\_dir }}/node-exporter.service"

- name: "start and enable service"

#  when: copy\_conf\_status.changed == true or copy\_status.changed == true

  service:

    name: node-exporter

    state: restarted

    enabled: yes

    daemon\_reload: yes

**templates/node-exporter.service.j2**

[Unit]

Description=Prometheus Node Exporter Service

After=network.target

[Service]

User={{ node\_exporter\_user }}

Group={{ node\_exporter\_user }}

Type=simple

ExecStart=/usr/local/bin/{{ node\_exporter\_user }}

[Install]

WantedBy=multi-user.target

**vars/main.yml**

node\_exporter\_user: node\_exporter

service\_dest\_dir: /etc/systemd/system

node\_exporter\_download\_dir: /tmp/node\_exporter-1.7.0.linux-amd64

node\_exporter\_download\_url: https://github.com/prometheus/node\_exporter/releases/download/v1.7.0/node\_exporter-1.7.0.linux-amd64.tar.gz

**Prometheus**

**handlers/main.yml**

- name: restart\_prometheus

  systemd:

    name: prometheus

    state: restarted

    enabled: yes

    daemon\_reload: yes

- name: Reload Prometheus

  uri:

    url: http://{{ inventory\_hostname }}:9090/  # Use a variable for hostname

    method: POST

    status\_code: 200

  register: prometheus\_reload  # Register the result

- name: Show error if Prometheus reload fails

  debug:

    msg: "Failed to reload Prometheus: {{ prometheus\_reload.msg }}"

  when: prometheus\_reload.status != 200

**task/main.yml**

- name: Create user for prometheus

  user:

    name: "{{ prometheus\_user }}"  # Username

    state: present  # Ensure the user is present

    createhome: no  # Create the user's home directory

    shell: /usr/sbin/nologin

    append: yes

  become: true

- name: Create installation folder in etc

  file:

    path: "{{ prometheus\_dest\_dir }}"

    state: directory

    owner: "{{ prometheus\_user }}"

    group: "{{ prometheus\_user }}"

    mode: '0775'

    recurse: yes

- name: Create data folder

  file:

    path: "{{ prometheus\_data\_dir }}"

    state: directory

    owner: "{{ prometheus\_user }}"

    group: "{{ prometheus\_user }}"

    mode: '0775'

    recurse: yes

- name: Download/unarchive Packages for prometheus

  unarchive:

    src: "{{ prometheus\_download\_url }}"

    dest: /tmp

    owner: "{{ prometheus\_user }}"

    group: "{{ prometheus\_user }}"

    remote\_src: yes

    creates: "{{ prometheus\_download\_dir }}"

- name: Copying the service binary for prometheus

  copy:

    src: "{{ prometheus\_download\_dir }}/{{ item }}"

    dest: /usr/local/bin/

    owner: "{{ prometheus\_user }}"

    group: "{{ prometheus\_user }}"

    mode: '0775'

    remote\_src: yes

  with\_items:

    - prometheus

    - promtool

- name: Copying the console binary

  copy:

    src: "{{ prometheus\_download\_dir }}/{{ item }}"

    dest: "{{ prometheus\_dest\_dir }}"

    owner: "{{ prometheus\_user }}"

    group: "{{ prometheus\_user }}"

    mode: '0775'

    remote\_src: yes

  with\_items:

    - consoles

    - console\_libraries

- name: Removing the tar file of prometheus

  file:

    path: "{{ prometheus\_download\_dir }}"

    state: absent

- name: Create prometheus systemd service file

  template:

    src: templates/prometheus.service.j2

    dest: "{{ service\_dest\_dir }}/prometheus.service"

- name: prometheus args

  template:

    src: templates/prometheus.j2

    dest: /etc/default/prometheus

    mode: 0644

    owner: root

    group: root

  notify: restart\_prometheus

- name: Copying the scrape config

  template:

    src: templates/prometheus.yml.j2

    dest: "{{ prometheus\_dest\_dir }}/prometheus.yml"

    owner: "{{ prometheus\_user }}"

    group: "{{ prometheus\_user }}"

    mode: '0775'

- name: "start and enable service"

#  when: copy\_conf\_status.changed == true or copy\_status.changed == true

  service:

    name: prometheus

    state: restarted

    enabled: yes

    daemon\_reload: yes

**templates/prometheus.yml.j2**

global:

  scrape\_interval: 15s

scrape\_configs:

  - job\_name: 'prometheus'

    static\_configs:

      - targets: ['localhost:9090']

  - job\_name: 'ec2'

    ec2\_sd\_configs:

      - region: 'us-west-2'

        access\_key: " #Add Access\_Key "

        secret\_key: " #Add Secret\_Key "

        port: 9100

    relabel\_configs:

      - source\_labels: [\_\_meta\_ec2\_tag\_Group]

        regex: 'private'

        action: keep

**vars/main.yml**

prometheus\_user: prometheus

prometheus\_dest\_dir: /etc/prometheus

prometheus\_data\_dir: /var/lib/prometheus

service\_dest\_dir: /etc/systemd/system

prometheus\_download\_dir: /tmp/prometheus-2.51.0.linux-amd64

prometheus\_download\_url: https://github.com/prometheus/prometheus/releases/download/v2.51.0/prometheus-2.51.0.linux-amd64.tar.gz

prometheus\_dir\_configuration: "/etc/prometheus"

prometheus\_retention\_time: "365d"

prometheus\_scrape\_interval: "30s"

prometheus\_node\_exporter: true

prometheus\_node\_exporter\_group: "tags\_Group\_private"

prometheus\_env: "production"

prometheus\_var\_config:

  global:

    scrape\_interval: "{{ prometheus\_scrape\_interval }}"

    evaluation\_interval: 5s

    external\_labels:

      env: '{{ prometheus\_env }}'

  scrape\_configs:

    - job\_name: prometheus

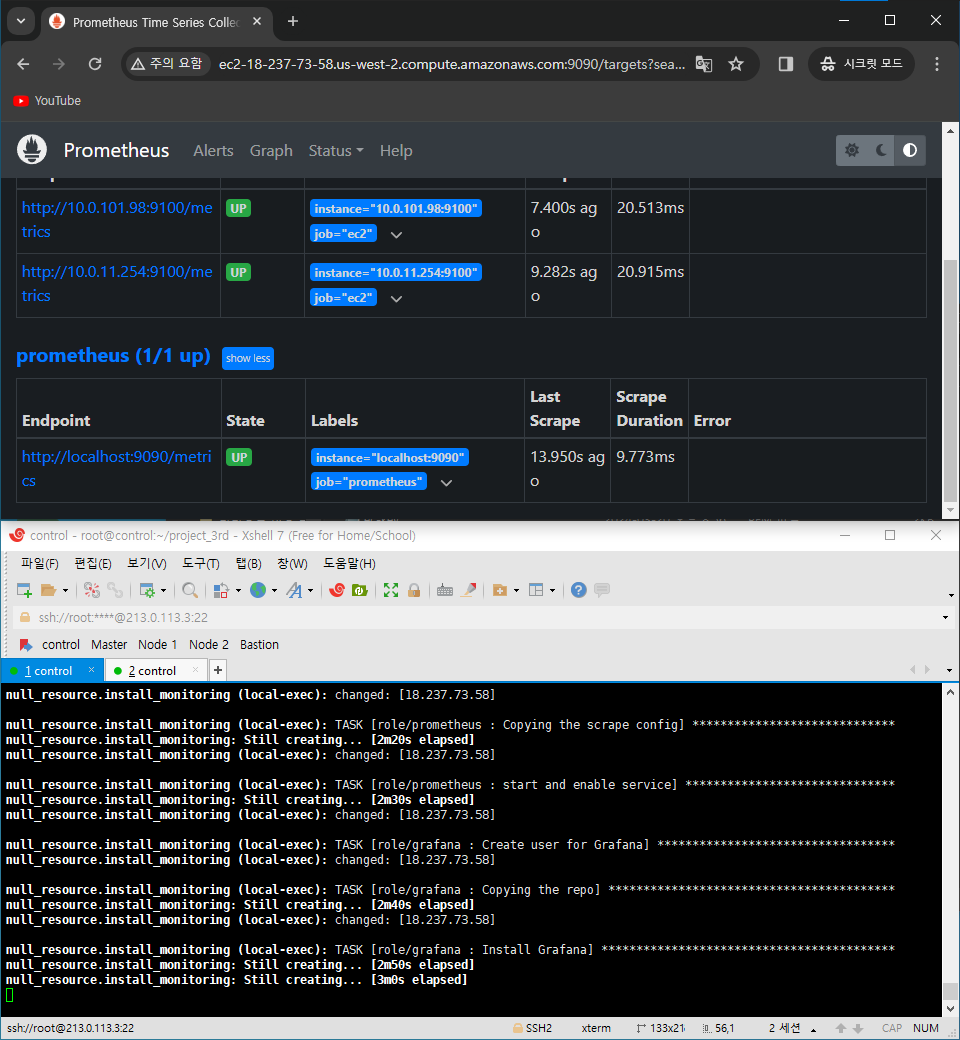
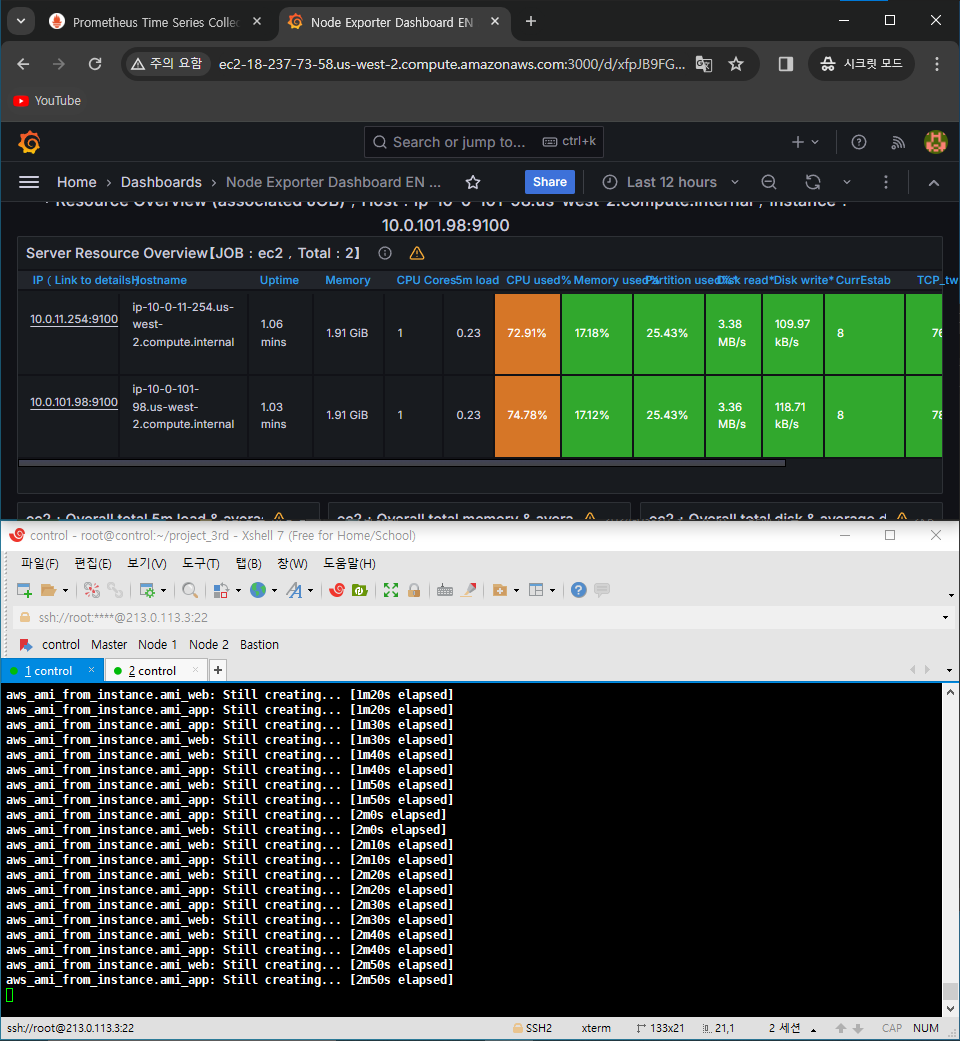
      scrape\_interval: 5m

      static\_configs:

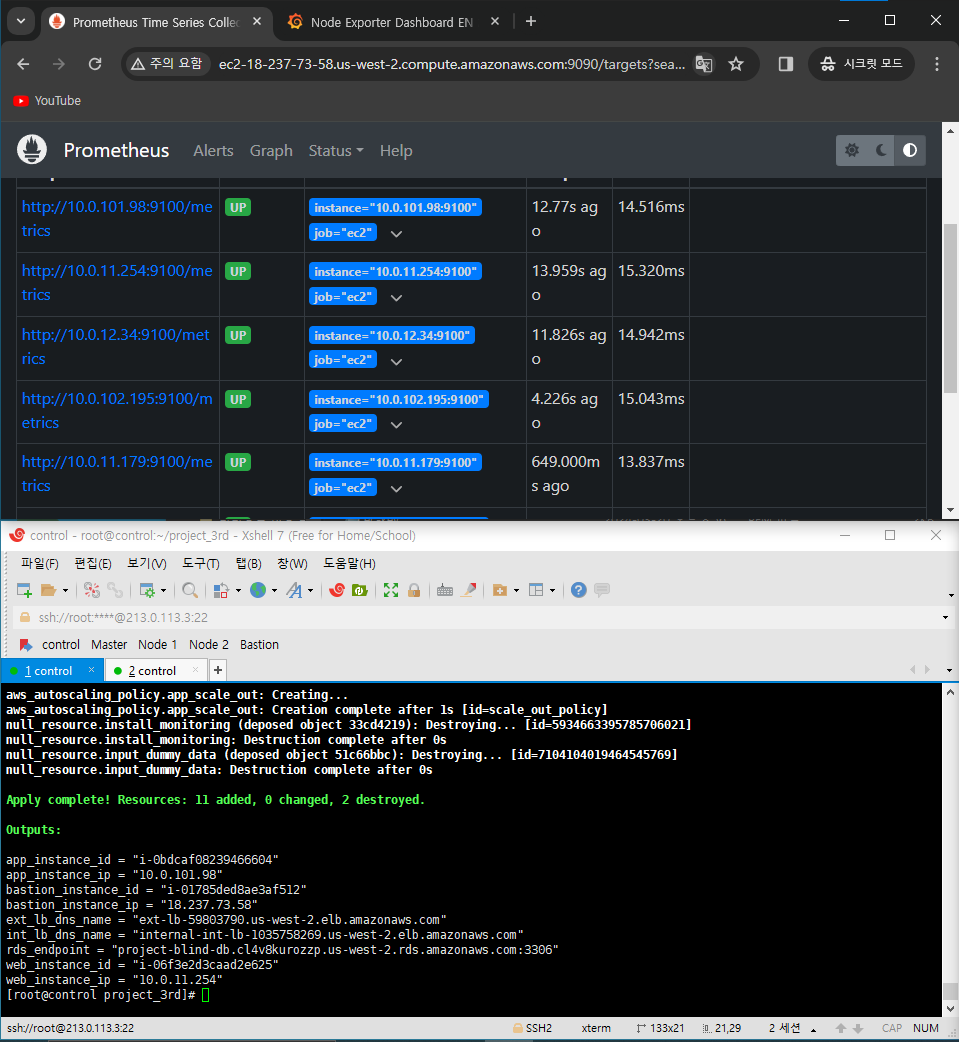
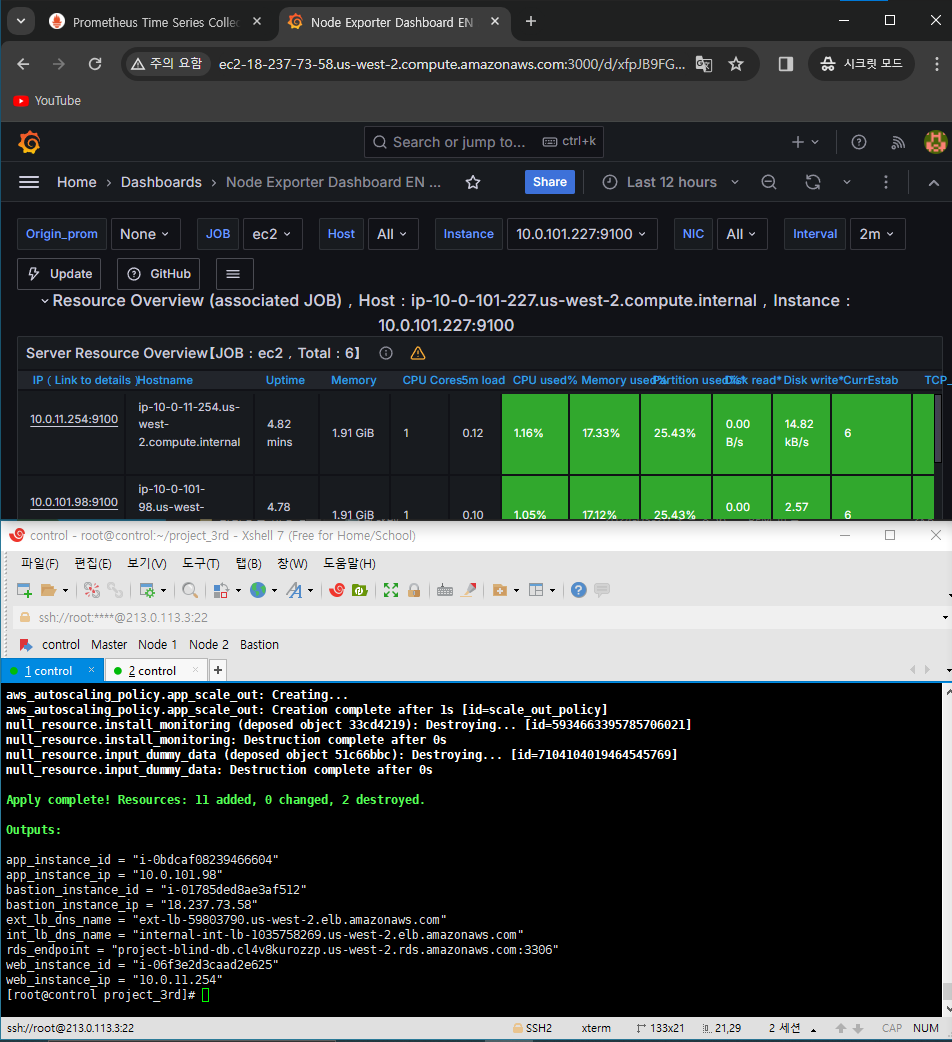
        - targets: ['{{ inventory\_hostname }}:9090']

**결과**

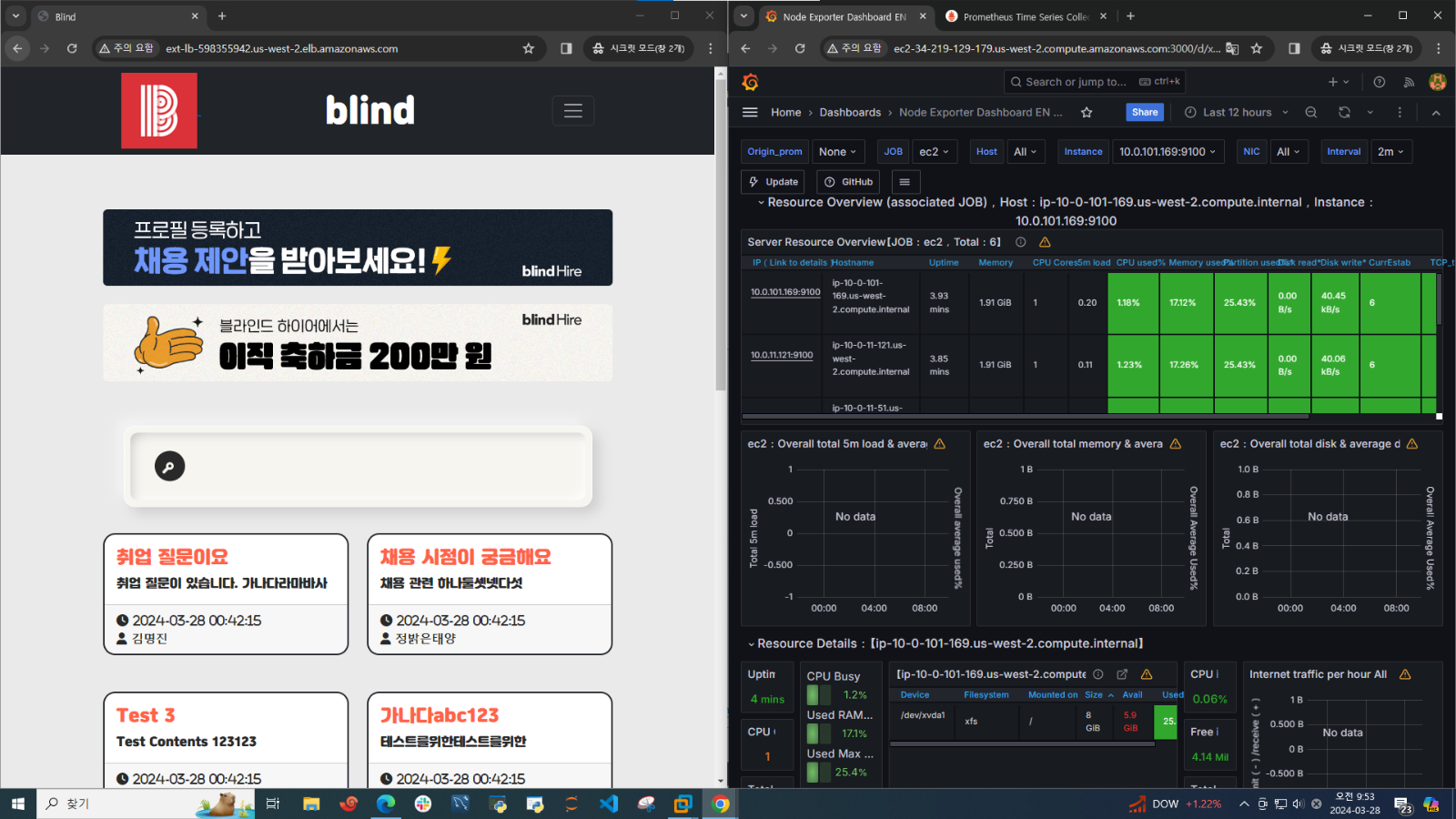
**AMI 생성 전 Prometheus & Grafana 실행**



**AMI 로 ASG 생성 후 Prometheus & Grafana 실행**



**최종 결과**

****

**[ 차후 발전 방향 및 소감 ]**

**정밝은태양**

**- 소감**

AWS 웹 페이지에서 했던 작업들을 Code화 하여 작업하는 방식을 이해 할 수 있었습니다. 또한, 팀원들간의 협업을 Code를 이용하여 보다 원할하게 프로젝트를 진행 할 수 있었습니다.

**- 아쉬운 점**

Test Bed 용 Web & WAS Instance를 Private Subnet에 생성하여, Service 파일을 Bastion Instance를 통해 넘겨주는 과정에서 제약 되는 점이 있어 많은 시간이 소비되었습니다. 초기 계획 하였던, EKS를 활용한 서비스 제공을 완성 시키지 못한 점이 아쉬웠습니다.

**김명진**

**- 소감**

Ansible만 사용하여 모든 과제를 해결하려고 했으나 인프라 구축에서 많은 애로사항이 있어 시간이 많이 지연되었습니다. 특히, Inventory를 통해 서비스를 실행하는 과정에서 대상을 지정하는 부분에서 많은 어려움을 겪었습니다. 못다했던 부분은 추가적으로 공부를 해서 완성하고 싶습니다.

**남정현**

**- 소감**

콘솔에서 리소스를 만들때보다 자원관리에있어서 자동화기능을 통해 편리함을 느꼈습니다.  
또한, 테라폼의 state상태파일을 통해 기존인프라와 비교하여 변경사항이없으면 넘어가는 테라폼의 멱등성 특징을 몸소 체험할 수 있었습니다.

**우희제**

**- 소감**

콘솔에서 클릭만 해서 만들다가 코드로 하나하나 구성하려니 기존에 쉽게 지나친 부분에서 막혀 오류가 발생하는 경우가 많았다. 앞으로는 좀 더 꼼꼼하게 조건을 확인해야겠다. 또한 이번에 시간 문제로 그다지 다룰 기회가 없었던 ansible은... 제대로 할 수 있을까?

정하늘

**- 소감**

3-tier 를 생성하면서 aws console 에서 수동으로 잡아주던 부분이 무엇을 의미하고,어떤 부분과 서로 연계하여 작동하는지 이해할 수 있는 시간이었습니다.작업의 전체적인 work flow 를 구상하고 그에 맞게 terraform resource 들이 작동할 수 있도록 서로간의 의존성을 확립하지 않으면 특정 작업이 일찍 혹은 늦게 시작 되어 전체적인 작업 완성 시간이 늘어질 수 있다는 것도 알게 되었습니다.Prometheus 와 Node-exporter 를 설치하면서 Dynamic Inventory 의 작동 방식과 사용 방법을 이해할 수 있었고, 특히 Prometheus 의 scrap target 을 설정하는 과정에서 jinja 의 for 문을 사용하여 반복문을 구성하고, magic variable 을 연습해볼 수 있었습니다.

**- 아쉬운점**

Ansible 을 이용해 monitoring 환경을 구축할 때 여러 다양한 변수를 사용하다 보니 문제가 많이 발생되었습니다.amazon 에서 제공하는 Amazon Managed Grafana 와Amazon Managed Service for Prometheus 가 있습니다. cloud watch를 이용해 node의 정보를 가져와 시각화 해주는 서비스였습니다. Terraform 에서는 aws\_prometheus\_\*\*\*\*, aws\_grafana\_\*\*\*\* 등의 resource를 이용해 적용 할 수 있다는 것을 알게 되었지만 적용해보지 못한 것이 아쉬웠습니다.

**[부록]**

<공통>

# 보안 그룹

|  |  |  |  |
| --- | --- | --- | --- |
| **이름** | **ingress source** | **port** | **describe** |
| bastion\_sg | 0.0.0.0/0 | 22 | ssh form outside |
|  | 0.0.0.0/0 | 3000 | grafana form outside |
|  | 0.0.0.0/0 | 9090 | prometheus form outside |
| web\_sg | bastion\_sg.id | 22 | ssh from bastion |
|  | e\_elb\_sg.id | 5000 | flask service from e\_elb |
|  | bastion\_sg.id | 9100 | node-exporter scrape |
| was\_sg | bastion\_sg.id | 22 | ssh from bastion |
|  | i\_elb\_sg.id | 5000 | service from e\_elb |
|  | bastion\_sg.id | 9100 | node-exporter scrape |
| e\_elb\_sg | 0.0.0.0/0 | 80 | http from outside |
| i\_elb\_sg | web\_sg | 5000 | flask service from web layer |
| db\_sg | bastion\_sg.id | 3306 | rds from bastion |
|  | web\_sg.id | 3306 | rds from web\_layer |
|  | was\_sg.id | 3306 | rds from was\_layer |
| web\_efs\_sg | web\_sg.id | 2049 | efs from web\_layer |
| was\_efs\_sg | was\_sg.id | 2049 | efs from was\_layer |

**<terraform>**

**# 인스턴스**

**data.aws\_ami**

- 등록된 AMI 가운데 원하는 AMI의 id를 알기 위해 사용

- 지역을 바꿔도 이 기능을 동일하게 사용할 수 있음

**aws\_key\_pair**

- 인스턴스 생성시에 필요한 키페어를 등록하는 기능을 제공

- 이번에는 ssh-keygen으로 생성한 키페어를 사용함

**aws\_instance**

- 인스턴스 생성

**# 네트워크**

**aws\_vpc**

- vpc 생성. 빈 칸으로 작성시 기본 vpc

**aws\_subnet**

- 서브넷 생성.

**aws\_route\_table**

- 라우팅 테이블을 생성

**aws\_route\_table\_association**

- 라우팅 테이블과 서브넷을 연결

**aws\_eip**

- 탄력적 ip 생성

**aws\_internet\_gateway**

- 인터넷 게이트웨이 생성

**aws\_nat\_gateway**

- nat 게이트웨이 생성

**aws\_security\_group**

- 보안 그룹 생성

**# 로드 밸런서**

**aws\_alb**

- 로드 밸랜서 생성

- internal 인수(bool)의 값에 따라 external과 internal 생성 가능

**aws\_lb\_target\_group**

- 대상 그룹 생성

**aws\_lb\_listener**

- 리스터를 생성

- 로드밸런서의 정보가 필요

**aws\_lb\_target\_group\_attachment**

- 대상 그룹에 인스턴스를 연결

**# EFS 파일 시스템**

**aws\_efs\_file\_system**

- efs 파일 시스템 생성

**aws\_efs\_mount\_target**

- efs 파일 시스템을 서브넷과 연결

**# DB**

**aws\_db\_instance**

- db 인스턴스 생성

**aws\_db\_subnet\_group**

- db 서브넷 생성

**# 오토 스케일링**

**aws\_ami\_from\_instance**

- 현재 존재하는 인스턴스의 AMI를 생성

**aws\_launch\_configuration**

- 오토 스케일링을 통해 생성될 인스턴스의 상세 정보를 담음

**aws\_autoscaling\_group**

- 오토 스케일링 그룹 생성

**# 권한**

**aws\_iam\_role**

- 역할 생성 및 관리

**aws\_iam\_policy**

- IAM 정책을 생성 및 관리

**aws\_iam\_policy\_attachment**

- IAM 정책을 역할 또는 사용자에게 첨부

**aws\_iam\_instance\_profile**

- 인스턴스에 대한 IAM 역할을 지정하는 역할

**aws\_autoscaling\_policy**

- Auto Scaling 그룹의 스케일링 동작을 제어

**aws\_cloudwatch\_metric\_alarm**

- Amazon CloudWatch 지표 알람을 나타내는 리소스

**aws\_sns\_topic**

- Amazon Simple Notification Service (SNS) 주제를 나타내는 리소스

**aws\_sns\_topic\_subscription**

- SNS 주제에 대한 구독을 생성 및 관리

**[ 코드 첨부 ]**

<https://github.com/WTGrape/blind_project_3rd>