terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "~>5.0"

    }

  }

}

provider "aws" {

  region = "us-east-1"

}

resource "aws\_instance" "instance" {

  ami = "ami-0c02fb55956c7d316"

  instance\_type = "t2.micro"

  key\_name = "devops"

  security\_groups = ["tf-provisioner-sg"]

  tags = {

    Name = "terraform-instance-with-provisioner"

  }

#local de bi şey yazmak için command

  provisioner "local-exec" {

      command = "echo http://${self.public\_ip} > public\_ip.txt"

  }

  connection {

    host = self.public\_ip

    type = "ssh"

    user = "ec2-user"

    private\_key = file("~/devops.pem")     # key.pem

  }

  provisioner "remote-exec" {

    inline = [ #Remote bir şey yüklemek için inline yaz

      "sudo yum -y install httpd",

      "sudo systemctl enable httpd",

      "sudo systemctl start httpd"

    ]

  }

  provisioner "file" {

    content = self.public\_ip #Localden uzak makinaya bir şey göndermek için(content içiriğini oku diyor)

    destination = "/home/ec2-user/my\_public\_ip.txt"

  }

}

resource "aws\_security\_group" "tf-sec-gr" {

  name = "tf-provisioner-sg"

  tags = {

    Name = "tf-provisioner-sg"

  }

  ingress {

    from\_port   = 80

    protocol    = "tcp"

    to\_port     = 80

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

      from\_port = 22

      protocol = "tcp"

      to\_port = 22

      cidr\_blocks = [ "0.0.0.0/0" ]

  }

  egress {

      from\_port = 0

      protocol = -1

      to\_port = 0

      cidr\_blocks = [ "0.0.0.0/0" ]

  }

}

```

- Go to the Provisioners folder and run the terraform file.

```bash

terraform init

terraform apply

cat public\_ip.txt

ssh -i ~/sam.pem ec2-user@52.90.85.198

```

- Check the resources that created by terraform.

- Terminate the resources.

```bash

$ terraform destroy

ChatGPT

Bu Terraform kodu, AWS (Amazon Web Services) üzerinde bir EC2 (Elastic Compute Cloud) örnek oluşturmak ve bu örneği yapılandırmak için kullanılır. Aşağıda kodun her bir bölümünün ne yaptığını ve nasıl çalıştığını adım adım açıklayacağım:

1. **aws\_instance Kaynağı**: Bu bölüm, bir AWS EC2 örneği oluşturur ve yapılandırır. İşte bu kaynağın önemli özellikleri:
   * **ami**: Bu, örneğin hangi Amazon Machine Image (AMI) ile oluşturulacağını belirtir. Bu kod örneği, "ami-0c02fb55956c7d316" adlı bir Amazon Linux 2 AMI kullanır.
   * **instance\_type**: Oluşturulan örneğin türünü belirtir. Bu örnek, "t2.micro" türünde bir örnek oluşturur.
   * **key\_name**: Oluşturulan örnek için kullanılacak SSH anahtar çiftini belirtir. Bu örnekte "devops" adlı bir anahtar kullanılır.
   * **security\_groups**: Örneğe atanan güvenlik gruplarını belirtir. Bu örnekte, "tf-provisioner-sg" adlı bir güvenlik grubu kullanılır.
   * **tags**: Oluşturulan örneğe etiketler ekler. Bu örnekte, "Name" etiketi "terraform-instance-with-provisioner" olarak ayarlanır.
2. **local-exec Provisioner**: Bu provizyoner, örnek oluşturulduktan sonra yerel makinede bir komut çalıştırmak için kullanılır. Bu komut, örneğin genel IP adresini alır ve bir "public\_ip.txt" dosyasına yazdırır.
3. **connection Bloğu**: Bu bölüm, SSH ile örneğe bağlanmak için gerekli olan bağlantı bilgilerini sağlar. İşte bu bağlantı özelliklerinin anlamları:
   * **host**: SSH bağlantısının hedefi olarak örneğin genel IP adresini kullanır.
   * **type**: Bağlantı türünü belirtir (SSH).
   * **user**: SSH ile kullanılacak kullanıcı adını belirtir.
   * **private\_key**: SSH kimlik doğrulama için kullanılacak özel anahtar dosyasının yolu.
4. **remote-exec Provisioner**: Bu provizyoner, örneğe uzak komutlar göndermek için kullanılır. Bu örnekte, Apache HTTP sunucusunu kurar, başlatır ve yapılandırır.
5. **file Provisioner**: Bu provizyoner, örneğe dosyalar kopyalamak için kullanılır. Bu kod, örneğin genel IP adresini içeren bir dosyayı "/home/ec2-user/my\_public\_ip.txt" yoluna kopyalar.
6. **aws\_security\_group Kaynağı**: Bu kaynak, güvenlik grubunu tanımlar. Güvenlik grubu, gelen ve giden trafiği düzenlemek için kullanılır. Bu örnekte, HTTP (port 80) ve SSH (port 22) trafiğine izin veren bir güvenlik grubu oluşturulur.

Sonuç olarak, bu Terraform kodu AWS üzerinde bir EC2 örneği oluşturur, bu örneği yapılandırır (HTTP sunucusunu kurar), yerel makinede bazı işlemler yapar (genel IP adresini dosyaya yazdırır) ve güvenlik grubunu ayarlar. Bu sayede bir örnek oluşturup yapılandırarak AWS altyapınızı otomatik olarak yönetebilirsiniz.

host = self.public\_ip burdaki kimin puclic\_ip terraform makinasının mı terraform makinası aracılığıyla oluşturlan makinanın mı

ChatGPT

**host = self.public\_ip** ifadesi Terraform kodunda kullanılan bir dize içindedir. Bu ifade, Terraform tarafından yorumlanır ve **self.public\_ip** ifadesi, Terraform tarafından yönetilen kaynağın genel IP adresini temsil eder. Yani, bu ifade örneğin genel IP adresini ifade eder.

Örneğin, bu kod parçası bir AWS EC2 örneği oluşturuyorsa, **self.public\_ip** ifadesi, oluşturulan EC2 örneğinin genel IP adresini temsil eder. Bu nedenle, **host = self.public\_ip** ifadesi, SSH bağlantısının bu örneğin genel IP adresine yönlendirilmesini sağlar.

Kısacası, **self.public\_ip** ifadesi, Terraform tarafından yönetilen kaynağın genel IP adresini ifade eder ve bu bağlamda, AWS EC2 örneği için ilgili örneğin genel IP adresini temsil eder.

provisioner "file" { content = self.public\_ip destination = "/home/ec2-user/my\_public\_ip.txt" } bu ne iin

ChatGPT

Bu kod parçası, Terraform ile oluşturulan AWS EC2 örneğine ait genel IP adresini alır ve bu IP adresini örneğin içindeki **/home/ec2-user/my\_public\_ip.txt** dosyasına kaydeder. Bu işlem, örneğin genel IP adresini bir dosyada saklamak veya başka bir işlemde kullanmak için yapılır.

Özelliklerin açıklamaları:

* **content**: Bu özellik, dosyaya yazılacak içeriği temsil eder. Bu durumda, **self.public\_ip** ifadesi, örneğin genel IP adresini ifade eder.
* **destination**: Bu özellik, kaydedilecek dosyanın yolunu belirtir. Kod, örneğin içinde **/home/ec2-user/my\_public\_ip.txt** yoluna bu IP adresini kaydeder.

Sonuç olarak, bu kod parçası oluşturulan EC2 örneğinin genel IP adresini alır ve bu adresi **/home/ec2-user/my\_public\_ip.txt** dosyasına kaydeder, böylece ileride bu IP adresine başka işlemlerde veya uygulamalarda ihtiyaç duyulduğunda kullanılabilir.

provisioner "local-exec" { command = "echo http://${self.public\_ip} > public\_ip.txt" } burada local-exec değiştirilebilir mi

ChatGPT

Evet, **provisioner "local-exec"** ile belirtilen yerel komut değiştirilebilir. Bu bölüm, Terraform ile yerel makinenizde çalıştırılacak bir komutu temsil eder. Bu komutu değiştirmek isterseniz, **"command"** özelliğini değiştirerek yeni bir komut tanımlayabilirsiniz.

Örneğin, **echo http://${self.public\_ip} > public\_ip.txt** komutu, örneğin genel IP adresini **public\_ip.txt** adlı bir dosyaya yazdırmak için kullanılır. Eğer farklı bir işlem yapmak isterseniz, **"command"** özelliğini istediğiniz komutla değiştirebilirsiniz. Örneğin, özel bir betik çalıştırmak, veri çekmek veya başka bir yerel işlem yapmak için bu bölümü değiştirebilirsiniz.

peki niye yazdırıyor

ChatGPT

**provisioner "local-exec"** bölümündeki komut, örneğin genel IP adresini **public\_ip.txt** adlı bir dosyaya yazdırmak için kullanılıyor. Bu, oluşturulan EC2 örneğinin genel IP adresini bir dosyada saklamak ve bu IP adresine daha sonra erişmek veya kullanmak için yapılan bir işlemdir. Bu tür bir işlem, ağ yapılandırması veya otomasyon senaryolarında IP adresine ihtiyaç duyulduğunda faydalı olabilir.

Bu şekilde yapılan işlem, EC2 örneği oluşturulduktan sonra, örneğin genel IP adresinin nerede bulunabileceğini belirlemek için yerel makine üzerinde bir kaynak oluşturur. Bu kaynak sayesinde IP adresine daha sonra erişmek veya uygulamak için gerektiğinde bu dosyaya bakabilirsiniz.

command ne yapıyor

ChatGPT

**command** özelliği, **provisioner "local-exec"** bölümünde kullanılan bir özelliktir ve bu özellik ile belirtilen komut yerel (local) makinenizde çalıştırılır.

Örnekte verilen komut şu şekildedir:

hclCopy code

provisioner "local-exec" { command = "echo http://${self.public\_ip} > public\_ip.txt" }

Bu komut, **${self.public\_ip}** ifadesiyle örneğin genel IP adresini alır ve bu IP adresini "public\_ip.txt" adlı bir dosyaya yazar. Yani, bu komut, örneğin genel IP adresini alarak bu adresi bir metin dosyasına yazdırır.

Sonuç olarak, **command** özelliği, yerel makinenizde çalışacak olan belirli bir komutu temsil eder ve bu örnekteki komut, örneğin genel IP adresini bir dosyaya yazdırmak için kullanılır.

command yazmasaydık ne olurdu

ChatGPT

Eğer **command** özelliği belirtilmezse veya bu bölüm tamamen kaldırılırsa, **provisioner "local-exec"** olarak belirtilen yerel komut çalıştırılmaz. Bu durumda, Terraform, yerel makinede herhangi bir işlem gerçekleştirmez ve örneğin genel IP adresini yerel bir dosyaya yazmaz.

Yani, **command** özelliği belirtilmezse, bu provizyoner bloğu hiçbir işlem yapmaz ve herhangi bir çıktı üretmez. Bu, yerel işlemler veya konfigürasyon değişiklikleri yapmak istemediğiniz durumlarda kullanışlı olabilir.

provisioner ile başka hangi komutlar var

ChatGPT

Terraform'da **provisioner** ile birkaç farklı türde komut belirleyebilirsiniz. İşte yaygın olarak kullanılan bazı **provisioner** türleri ve bunların kullanım amaçları:

1. **local-exec**: Bu provizyoner, Terraform'ın çalıştığı yerel (local) makinenizde komutları çalıştırmak için kullanılır. Örneğin, dosya oluşturma, yerel betikleri çalıştırma veya yerel kaynakları yönetme gibi işlemler için kullanılabilir.

hclCopy code

provisioner "local-exec" { command = "echo 'Hello, Terraform'" }

1. **remote-exec**: Bu provizyoner, oluşturulan uzak örneğe SSH veya WinRM (Windows için) üzerinden komutlar göndermek için kullanılır. Genellikle sunucu yapılandırması veya uygulama dağıtımı gibi işlemler için kullanılır.

hclCopy code

provisioner "remote-exec" { inline = [ "sudo apt-get update", "sudo apt-get install -y nginx", ] }

1. **file**: Bu provizyoner, yerel makineden veya yerel bir kaynaktan dosyaları uzak bir örneğe veya yerel makineden uzak bir örneğe kopyalamak için kullanılır.

hclCopy code

provisioner "file" { source = "local-path/file.txt" destination = "/remote-path/file.txt" }

1. **chef, puppet, ansible, vs.**: Terraform, bazı yapılandırma yönetim araçlarıyla entegre olabilen özel provizyonerler sağlar. Bu provizyonerler, ilgili yapılandırma yönetim aracını kullanarak uzak örneği yapılandırmak için kullanılır.

hclCopy code

provisioner "chef" { # Chef yapılandırma komutları burada belirtilir. }

1. **null-resource**: Bu özel provizyoner, özel bir işlem veya komut çalıştırmak için kullanılır. Null-resource, Terraform kaynaklarına bağlı olmayan bağımsız bir provizyonerdir.

hclCopy code

resource "null\_resource" "example" { triggers = { # Değişkenlerin değerlerini burada belirtin } provisioner "local-exec" { command = "echo 'Custom provisioner'" } }

Bu provizyoner türleri, Terraform ile farklı senaryolara ve gereksinimlere uyacak şekilde kullanılabilir. İhtiyacınıza göre uygun olanını seçebilir ve özelleştirebilirsiniz.

burada illa localde dosya yazdırmak için "local-exec" ifadesi olmalı mı başka ifade de olabilir mi

ChatGPT

Hayır, bir kaynağın provizyonlanması için illa ki **local-exec** kullanmanız gerekmez. **local-exec**, yerel (local) makinede komutların çalıştırılması için kullanılan bir provizyoner türüdür, ve bu provizyonerle yerel işlemler gerçekleştirebilirsiniz. Ancak, kaynakları yönetmek veya yapılandırmak için diğer provizyoner türlerini kullanabilirsiniz.

Başka bir provizyoner türü kullanarak dosya yazdırabilirsiniz. Örneğin, aşağıdaki gibi **remote-exec** provizyoneri kullanarak uzak bir örnekte dosya oluşturabilirsiniz:

hclCopy code

provisioner "remote-exec" { inline = [ "echo 'http://${self.public\_ip}' > public\_ip.txt", ] }

Bu provizyoner, oluşturulan uzak örnekte belirtilen komutu çalıştırarak bir dosyayı oluşturur. Dosyanın içeriği **echo** komutu ile ayarlanır ve uzak örneğin genel IP adresi kullanılarak dosya içine yazılır.

Hangi provizyoner türünü kullanacağınız, yapmak istediğiniz işlemlere ve senaryonuza bağlı olarak değişir. Dosya yazdırmak gibi basit bir işlem için **local-exec** veya **remote-exec** kullanabilirsiniz, ancak daha karmaşık işlemler veya yapılandırmalar için diğer provizyoner türleri daha uygun olabilir.

inline ne demek burada

ChatGPT

**inline**, Terraform'da kullanılan bir provizyoner özelliğidir ve bu özellik ile birlikte verilen komutların veya komut dizisinin bir liste olarak belirtilmesini sağlar. Bu komutlar sırayla çalıştırılır. Yani, **inline** özelliği ile belirttiğiniz komutlar, provizyoner çalıştırıldığında sırasıyla çalıştırılır.

Örneğin, aşağıdaki **remote-exec** provizyonerinde **inline** özelliği kullanılmıştır:

hclCopy code

provisioner "remote-exec" { inline = [ "echo 'http://${self.public\_ip}' > public\_ip.txt", "touch example.txt", ] }

Bu örnekte, **inline** özelliği altında iki komut bulunmaktadır. İlk komut, bir dosyaya yazma işlemi gerçekleştirir ve örneğin genel IP adresini **public\_ip.txt** adlı bir dosyaya yazar. İkinci komut ise basitçe **example.txt** adlı bir dosya oluşturur.

**inline** özelliği, provizyoner tarafından çalıştırılacak komutları kolayca bir liste halinde belirlemenize olanak tanır ve bu komutlar sırayla çalıştırılır.

inline yazmaksak ne our

ChatGPT

Terraform'da bir provizyoner kullanırken **inline** özelliğini kullanmazsanız, provizyonerin ne yapması gerektiğini belirten komutlar yerine başka bir provizyoner özelliği veya belirtilen işlemin varsayılan davranışı kullanılır.

Provizonerlerin belirli bir işlemi gerçekleştirmesi için bazı özelliklere ihtiyaç duyabileceği gibi (örneğin, **remote-exec** için **inline** ile komutları belirtmek), bazı provizyonerler varsayılan davranışa sahip olabilirler. Bu, provizyoner türüne ve kullanılan kaynağa bağlı olarak değişebilir.

Örneğin, **local-exec** provizyonerinde **inline** kullanmadan yalnızca **command** özelliğini belirtirseniz, belirtilen komutun varsayılan olarak yerel makinenizde çalıştırılmasını beklersiniz:

hclCopy code

provisioner "local-exec" { command = "echo 'Hello, Terraform'" }

Bu komut, Terraform'ın çalıştığı yerel makinenizde varsayılan bir şekilde çalıştırılacaktır.

Provizonerlerin nasıl çalıştığını ve ne tür özelliklere ihtiyaç duyduğunu belirlemek için Terraform belgelerini ve kaynak belgelerini dikkatlice incelemeniz önemlidir, çünkü her provizyoner farklı davranışlar sergileyebilir ve farklı özelliklere ihtiyaç duyabilir.

provisioner "file" { content = self.public\_ip destination = "/home/ec2-user/my\_public\_ip.txt" } bu yerel makinaya mı uzak makinayı mı yazdırıyor

ChatGPT

Bu provizyoner, **provisioner "file"** kullanılarak belirtilen işlemi yerel makineden uzak bir makinaya gerçekleştirir. Yani, **self.public\_ip** ifadesi olan genel IP adresini, uzak bir EC2 örneğine belirtilen yol olan "/home/ec2-user/my\_public\_ip.txt" dosyasına yazdırmak için kullanılır.

Dolayısıyla, bu provizyoner, genel IP adresini yerel makineden uzak bir EC2 örneğine yazdırmak için kullanılır. Uzak makineye dosya kopyalama ve yazma işlemi bu provizyoner tarafından gerçekleştirilir.