**소개**

**Cloud Intensive 최종 시험을 위한 실습 가이드 라인입니다.**

**진행방식**

**3일 중 앞의 2일은 단체 실습 과제를 진행하며, 마지막 날은 오전 필기시험 후 오후 개인 실습 과제를 진행합니다.**

**필요한 것**

실습에 들어가기 전 **Terraform과 Git/Github, AWS/Azure 등에 대한 이해가 필요**합니다.

**Terraform이란?**

**코드로 클라우드 인프라를 관리할 수 있는 도구**.   
(Hashicorp사가 만든 HCL 언어를 이용한 Provisioning 도구)

**Git / GitHub란?**

**Git : 웹을 통한 분산 버전 관리**.

**-설치 :** [**https://git-scm.com/downloads**](https://git-scm.com/downloads)

**Github : Git을 저장할 수 있는 사이트. 쉽게 말해 코드 올려서 공유하는 사이트.**

**Repository : Github에서 실제로 Git을 저장하는 장소. 코드 버전이 보관되는 저장소.**

**Azure / AWS란?**

**Azure : Microsoft 사의 클라우드 서비스**.

**AWS : Amazon 사의 클라우드 서비스.**

**Putty / Puttygen이란?**

**Putty :** **SSH, 텔넷, rlogin, raw TCP를 위한 클라이언트로 동작하는 접속 프로그램**.

-설치 : <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

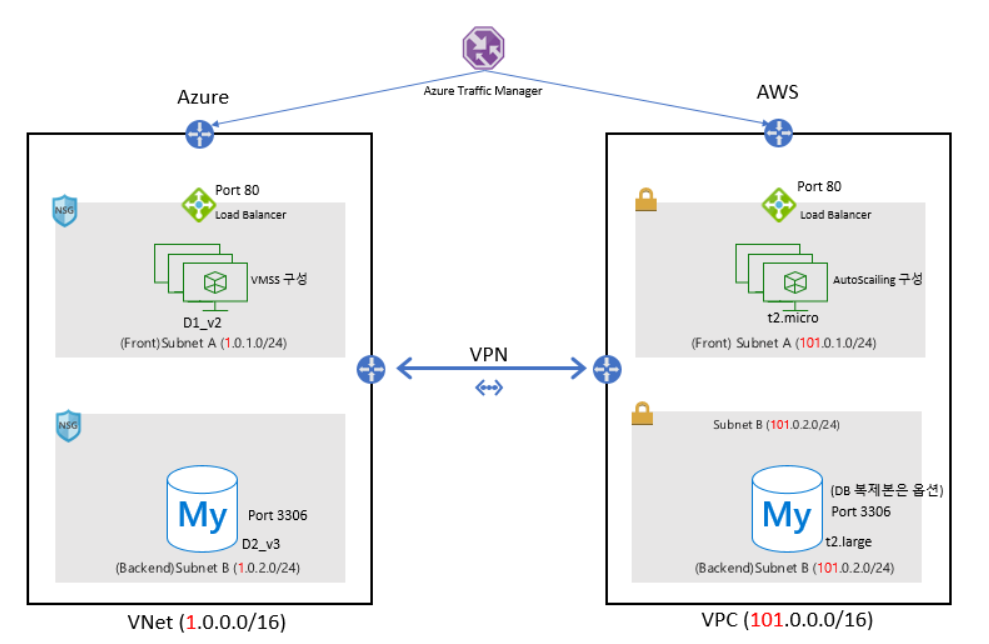
**Puttygen : 윈도우에서 SSH 키를 생성하기 위해 사용되는 프로그램.**

-설치 : [https://www.puttygen.com/#Download\_PuTTYgen\_on\_Windows](https://www.puttygen.com/%23Download_PuTTYgen_on_Windows)

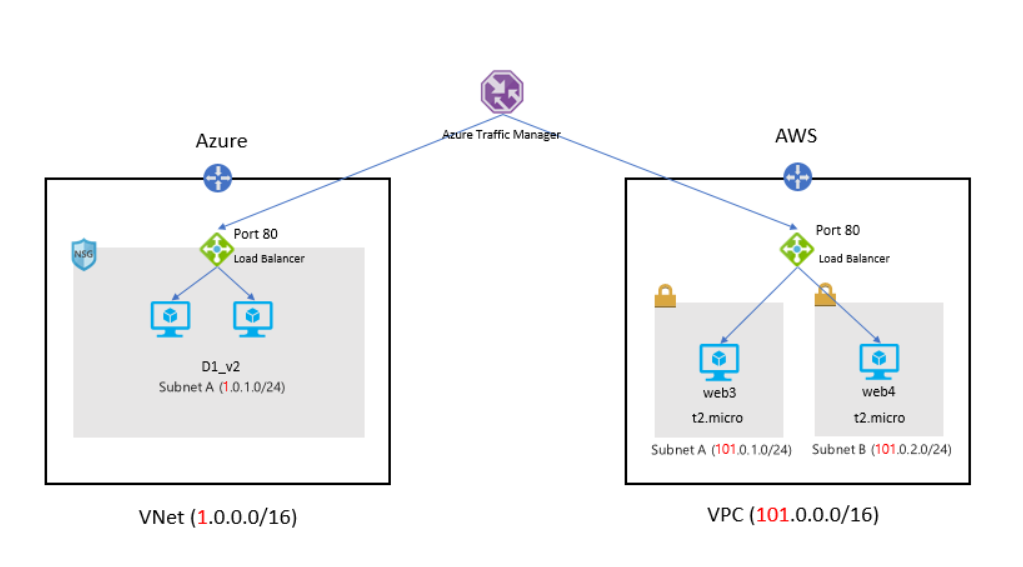
**위의 큰 개념을 복기, 설치하신 뒤 다음 그림을 보시겠습니다.**

**실습에서 만들어야 하는 최종 Output :**

**- 단체 실습**

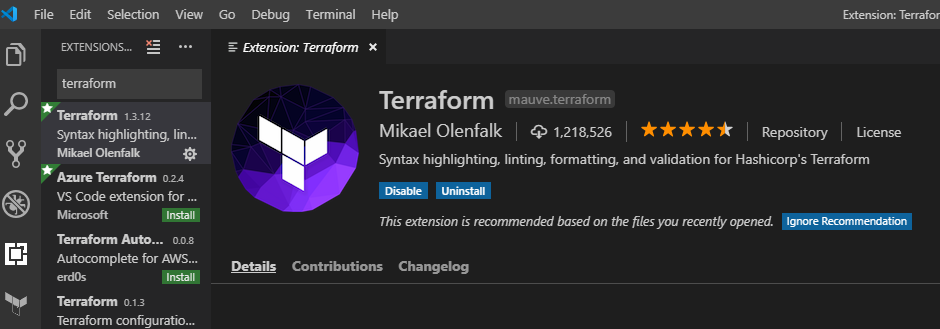


**- 개인 실습**



**1. 유저 별 할당 받은 IP대역으로 위의 그림을 Terraform으로 만들어야 합니다.  
2. 만들어진 VM에 SSH로 접속해서 웹 서버 설정을 하고 웹으로 접속 테스트가 되는 모습을 보여주시면 됩니다.  
3. 만든 개인/실습 과제를 Git을 이용하여 시험에서 제시된 Repository에 올려주시면 끝입니다.**

**이러한 OutPut을 만들기 위해서는 먼저…**

**Terraform 관련 툴 설치  
1. Terraform 개발을 위한 IDE 툴 설치(VSCODE) :** <https://code.visualstudio.com/> **2. VSCODE 실행 후 Terraform Extension 설치** **3. Terraform 설치 : 하단 링크 참조**<https://practice.hooniworld.io/entry/Terraform-With-Windows-%EC%84%A4%EC%B9%98>

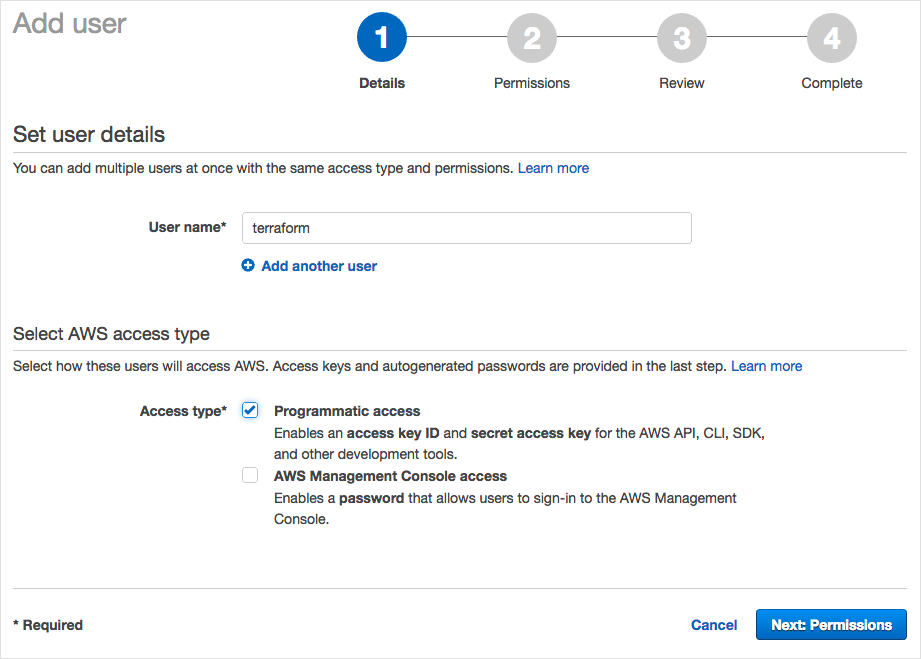
**4. Terraform 설치가 끝나면 cmd에서 terraform 명령 실행. 그러면 아래처럼 화면이 보입니다.**

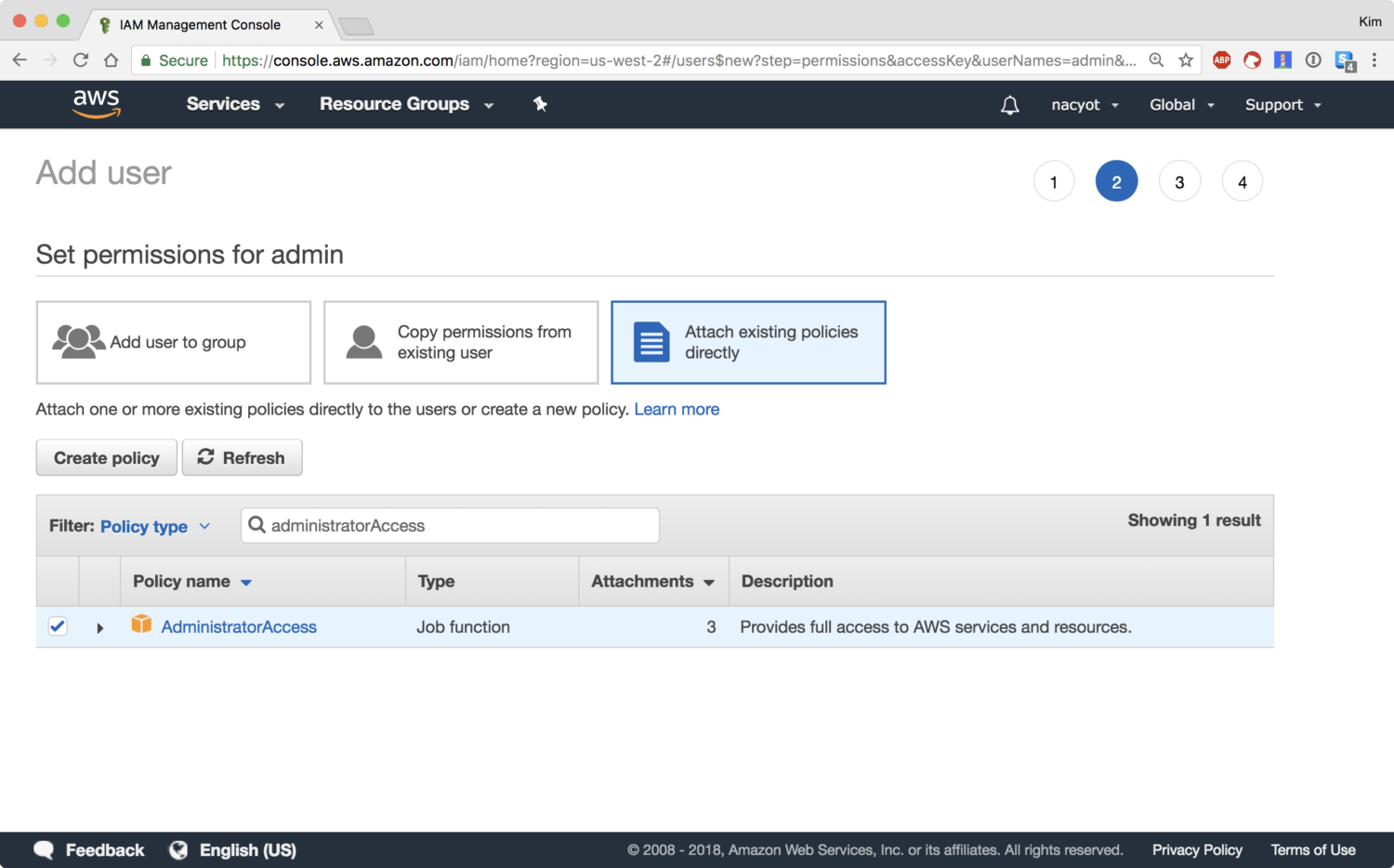


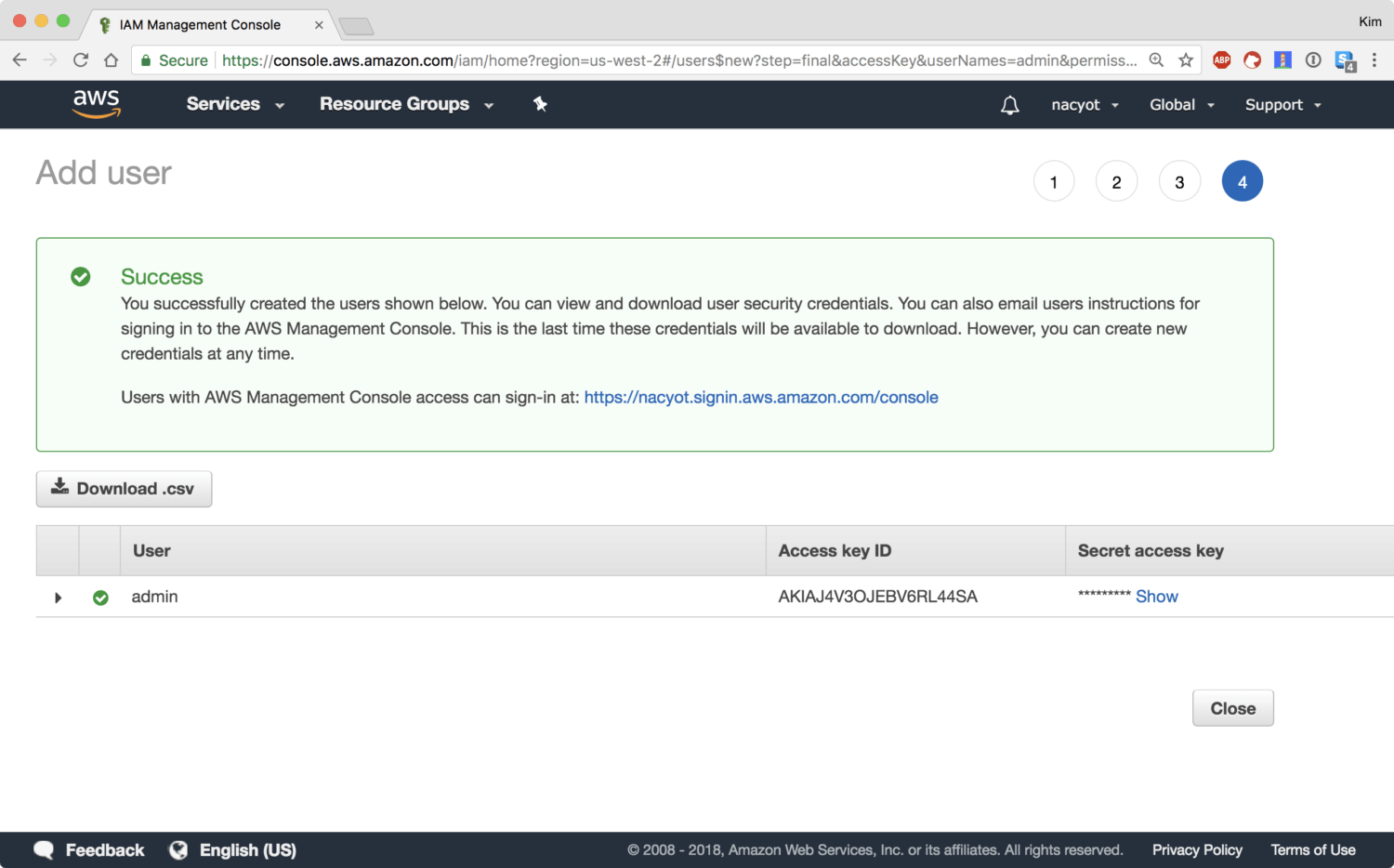
**5.** 위의 명령어들 중 **본 실습에 사용할 명령어는 세 가지이며,** 다음의 순서대로 사용합니다.  
**terraform init** 현재 폴더 내의 .tf 파일을 읽어 필요한 플러그인 로딩(Azurerm, AWS 등) **terraform plan** 현재 폴더 내의 .tf 파일을 읽어 코드 인프라 계획 수립  
**terraform apply** 계획된 대로 AWS/Azure 클라우드에 Resource 생성.

**6. 이제 Terraform을 위한 준비가 끝났습니다. 다음은 실제로 Terraform을 활용해 클라우드와 로컬을 연결해 보겠습니다.**

**Terraform으로 AWS/Azure에 접속하기 위한 준비**

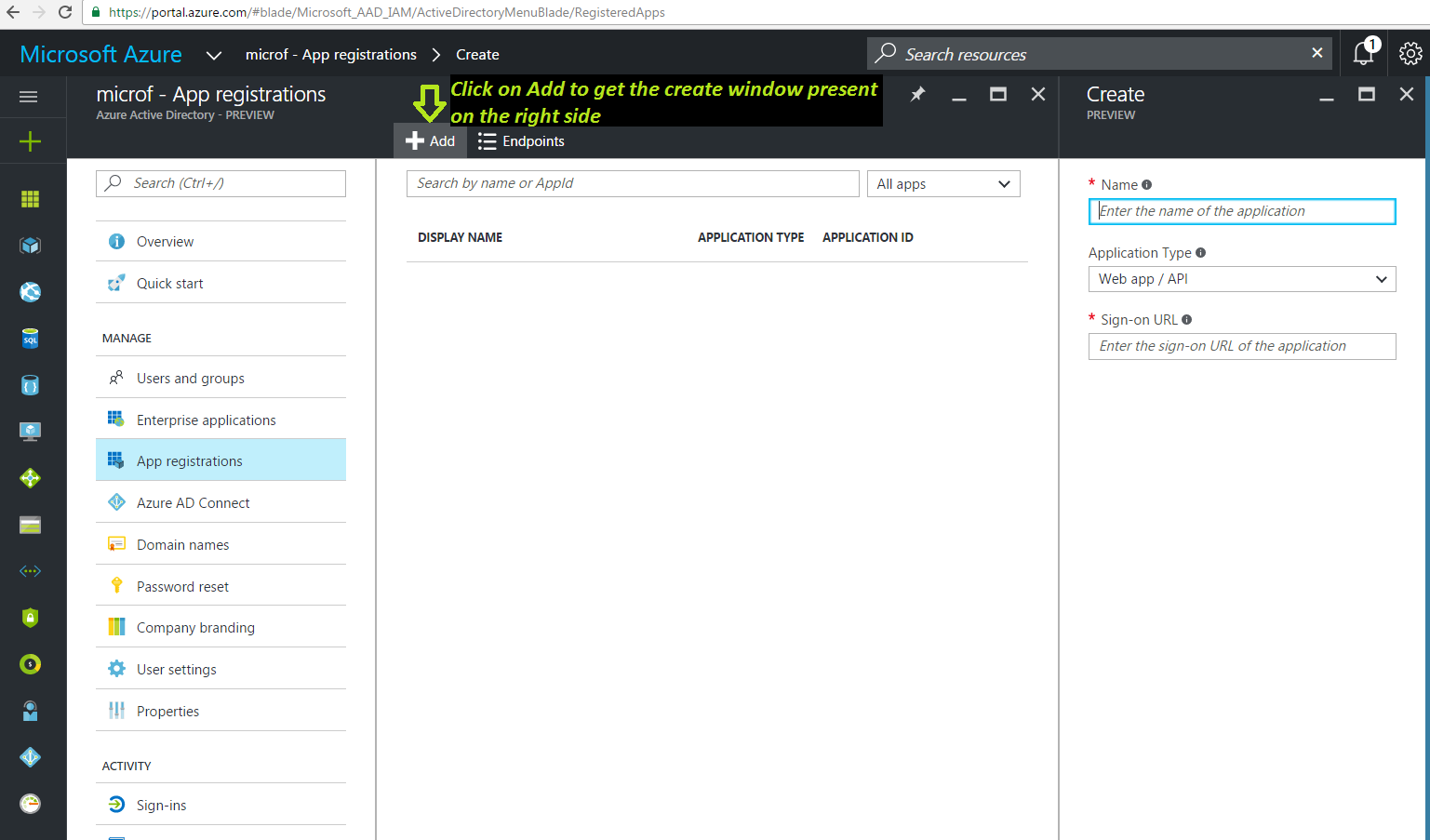
**1. AWS 매니지먼트 콘솔에서 Terraform을 위한 AWS 인증 키(IAM)를 생성  
 1-1.** [AWS Console IAM](https://console.aws.amazon.com/iam) – Users 메뉴에서 Add user를 선택하고 User를 생성.  
  **1-2. 필요한 권한(예제의 경우 Admin 권한)을 부여**

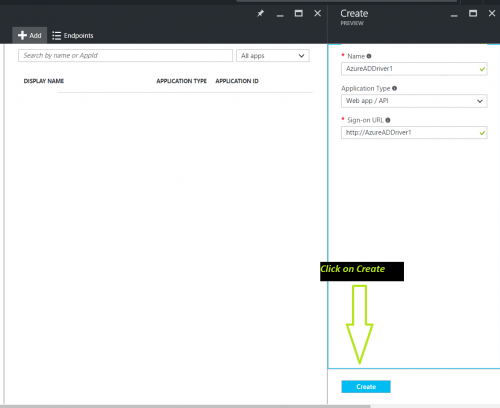


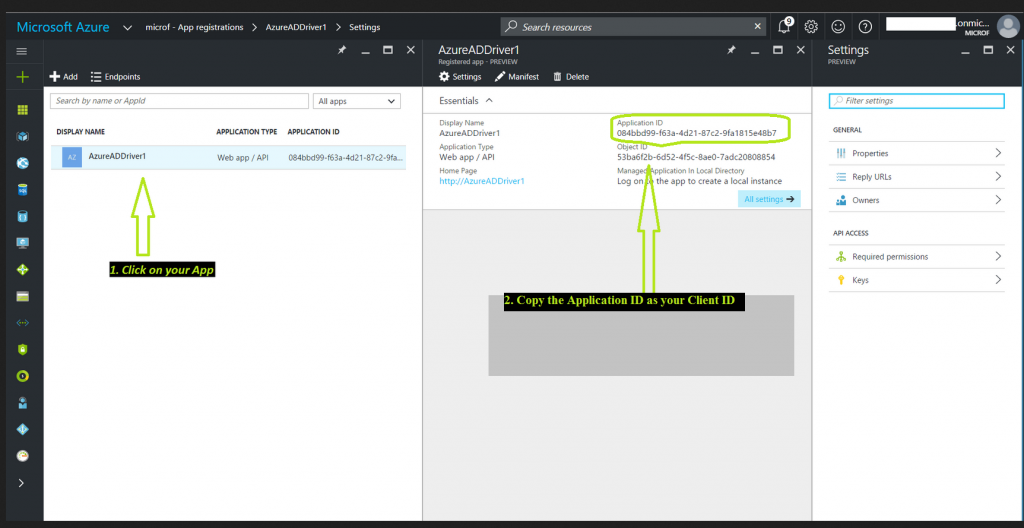
**1-3. 리뷰 후 생성하면 Terraform AWS를 위한 Access Key와 Secret Key 획득.**

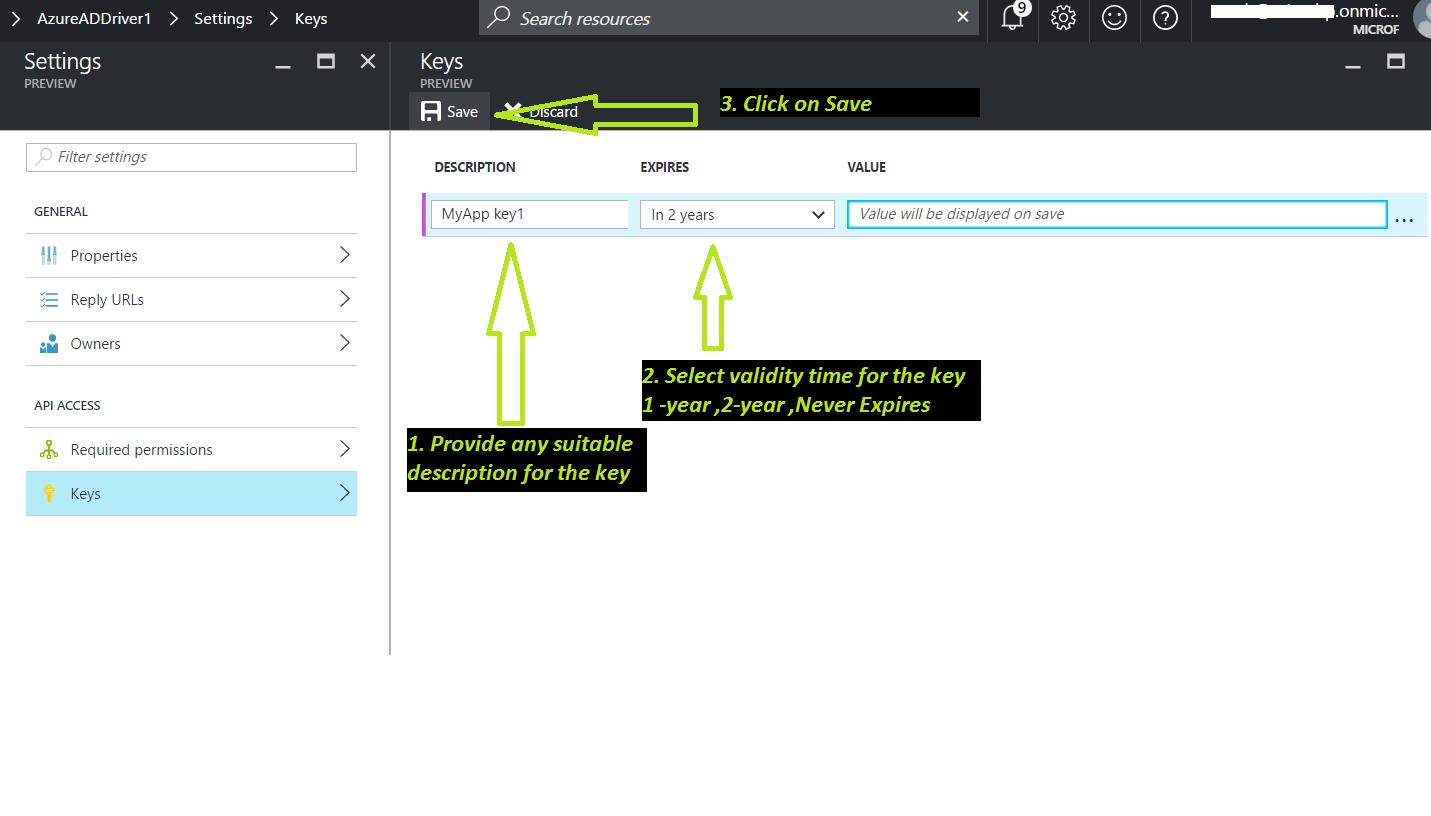
**2. Azure 포탈에서 Terraform을 위한 Azure 인증 키(IAM)를 생성**

**2-1. Active Directory 메뉴로 들어간 후 아래와 같이 앱 클릭**

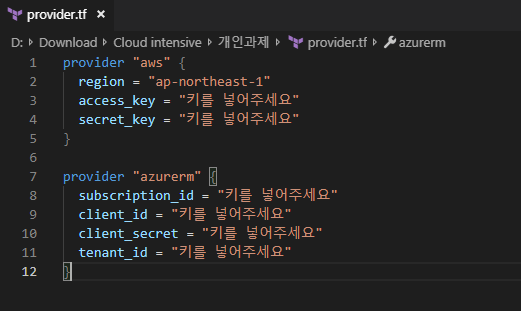
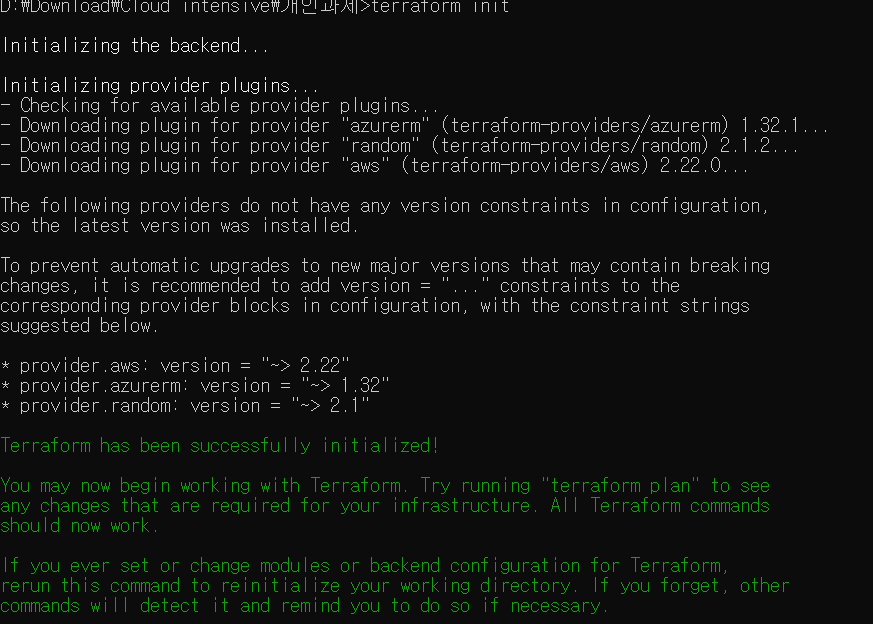


**2-2. create 클릭**

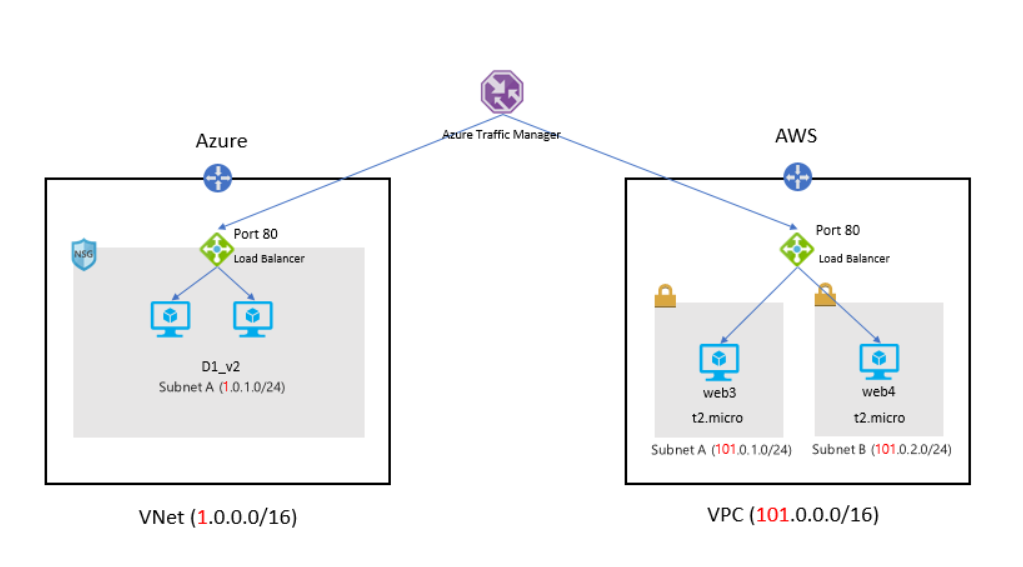
**2-3. Client ID 획득**

**2-4. 등록 후 시크릿 키 획득**

**\* 전체 과정은** <https://auth0.com/docs/connections/enterprise/azure-active-directory/v2> 참조

**3. 이렇게 얻은 키들을 VSCODE로 실습 폴더 Provider.tf 내에 정의합니다.  
3-1. Region, 각종 Key 등을 획득한 대로, 과제에서 제시해주는 대로 정의.**  **3-2. 이제 준비가 끝났습니다. cmd로 해당 폴더에서 Terraform init을 실행하면 정의된 provider를 가져옵니다.** 

**Terraform을 활용한 클라우드 구축(개인-기본)** 그럼 이제 **실제로 Terraform을 이용하여 실제 Resource들을 만들고 적용**해보겠습니다. 먼저 구조가 더 간단한 **개인 과제**를 바탕으로 설명 드리겠습니다.



그림만 보면 **Azure**에서 만들어줘야 하는 것은 **- VNet ({유저번호}.0.0.0/16)  
- Subnet 1({유저번호}.0.1.0/24)  
 - VM 1  
 - VM 2  
- LoadBalancer(port 80) 1개  
- Network Security Group**

**AWS**에서 만들어줘야 하는 것은 **- VPC 1개(1{유저번호}.0.0.0/16)  
- Subnet 1 (1{유저번호}.0.1.0/24)  
 - VM 1  
- Subnet 2 (1{유저번호}.0.2.0/24)  
 - VM 2  
- Application LoadBalancer(port 80) 1개  
- Security Group**

**입니다. 이후 AWS Traffic Manager를 정의하여 둘 간의 트래픽을 분산할 수 있도록 하는 것이 실습의 목표입니다.**

**기본적으로는 위의 것들만 정의 하시면 리소스가 만들어지는데요.**

**이를 Terraform 문법을 참고하여 차근차근 정의하고 구현하시면 됩니다.**

**먼저 간단하게 전체 과정을 설명 드리자면. 폴더 내에 *이름.*tf 파일을 만들어서**

variable "application\_port" {

description = "The port that you want to expose to the external load balancer"

default = 80

}

**식으로 사용할 변수 등을 지정하고**

resource "azurerm\_resource\_group" "rg" {

name = "user15-final"

location = "koreasouth"

}

**식으로 구현할 리소스를 지정하여 terraform으로 구현할 인프라를 정의합니다.**

backend\_port = "${var.application\_port}"

**인프라를 정의할 때 변수를 사용하시려면 위와 같이 진행하시면 됩니다.**

**이후**

1. **cmd에서 .tf 파일을 모아놓은 폴더로 접근, terraform init으로 provider별 플러그인 설치.**
2. **terraform plan으로 계획**
3. **terraform apply로 적용. yes입력 후 과정 관찰**

**을 통해 실제 AWS/Azure에서 선언한 리소스가 만들어지는 과정을 관찰할 수 있습니다.**

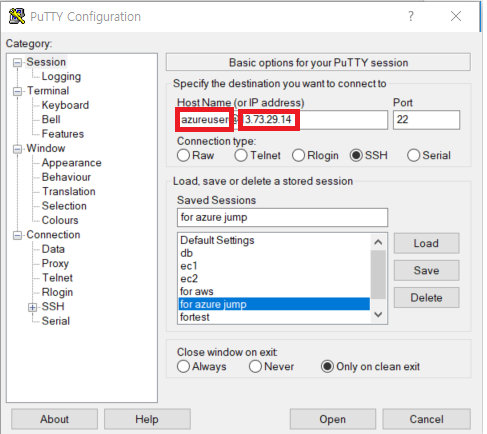
**이렇게 만들어진 리소스(VM)는 Windows OS 환경에서는 Putty를 활용해서 SSH로 접속할 수 있는데요.**

**AWS의 경우 : *VM에서 지정한 Username*@IP or DNS**

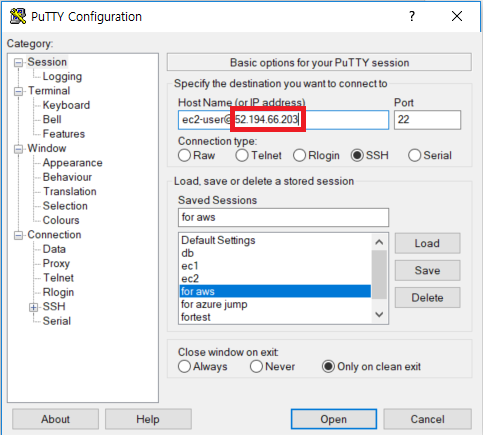
**Azure의 경우 : *ec2-user*@IP or DNS**

**의 호스트 네임으로 접속하시면 됩니다.**

**하단의 그림을 보시면 쉽게 이해하실 수 있습니다.**

****

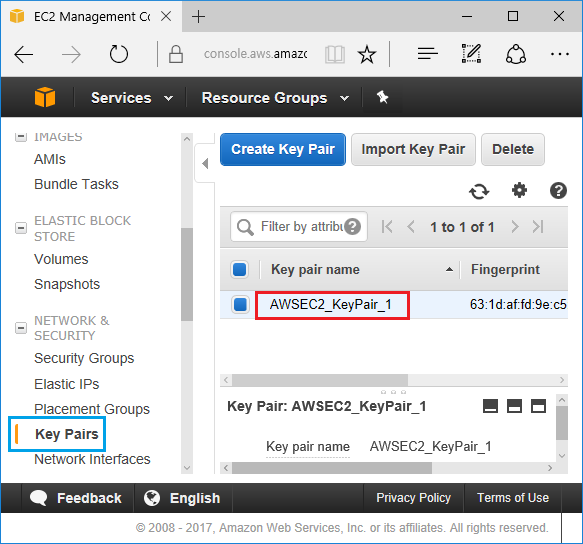
**<Azure VM 접속 방법 : azureuser는 임의로 지정한 값임.>**

****

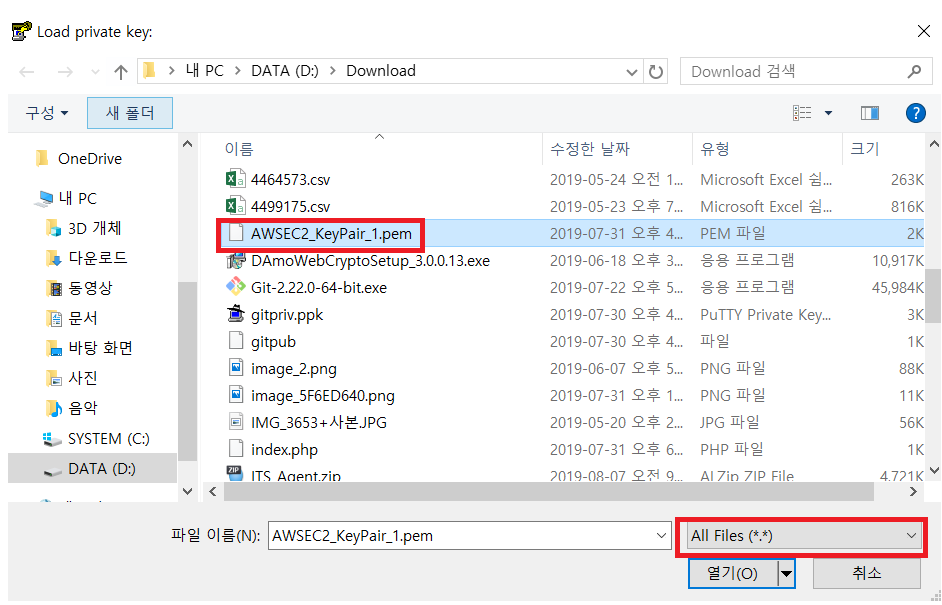
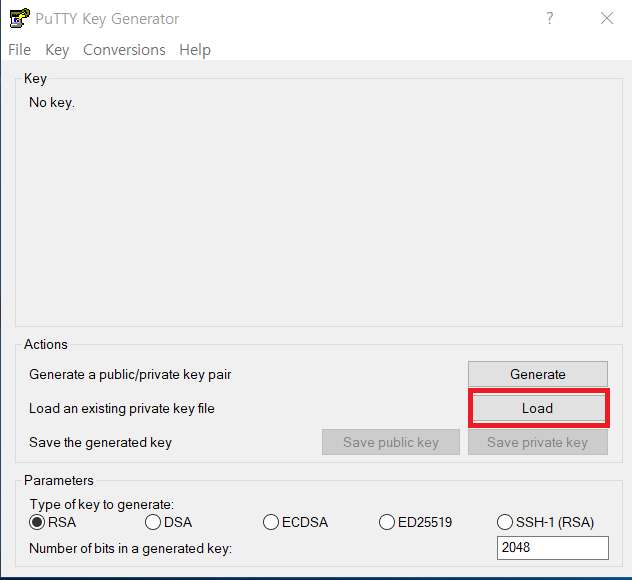
**<AWS VM 접속 방법 : ec2-user는 aws linux에서 고정으로 지정된 값임>**

**여기서 주의하실 점은 Terraform으로 Vm을 생성하기 전, 사용할 SSH Key Pair를 미리 생성해주셔야 접속이 원활하다는 점입니다.**

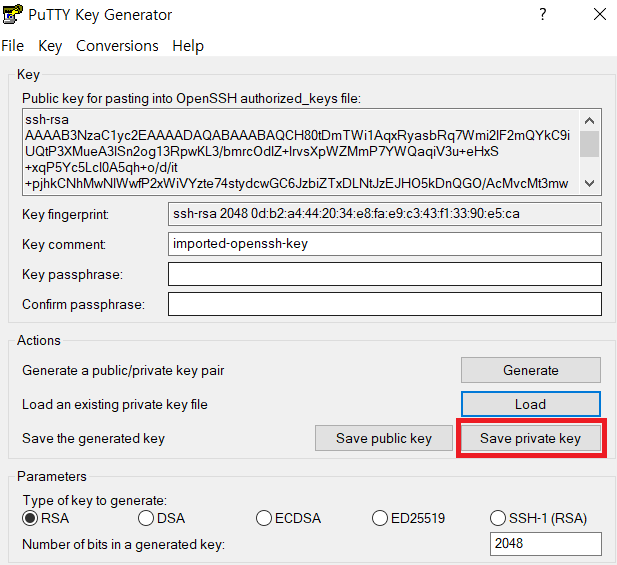
**예를 들어 AWS의 경우**



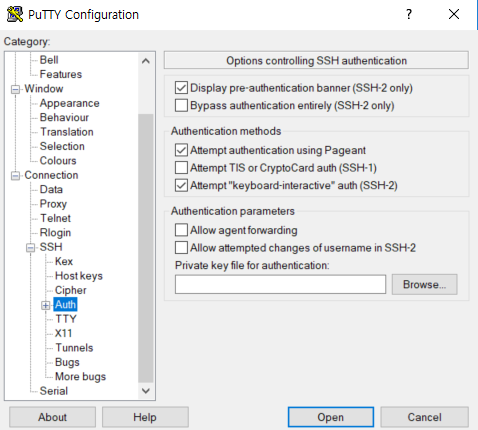
**위와 같이 해당 지역 내에서 사용할 Key-Pair를 먼저 생성해 주신 후. 생성과 동시에 다운되는 키 값을 받아 Puttygen을 활용해서 다음과 같이 Load하신 뒤**

****

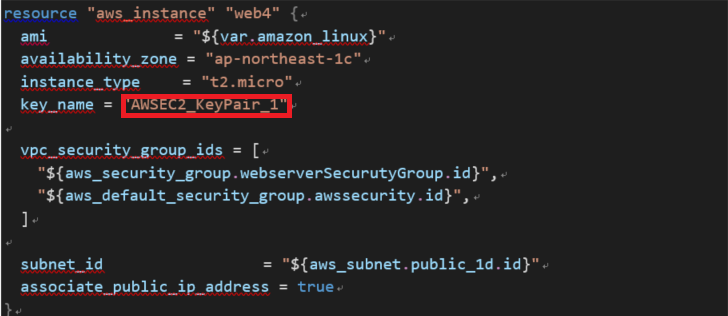
**다음처럼 Save Private Key를 활용해 Putty에 활용되는 .ppk 파일로 저장해주시고**

****

**이렇게 생성된 .ppk 파일을 다음과 같이 Connection-SSH-Auth 메뉴에서 지정해주셔야 해당 키를 활용한 VM과의 접속이 가능해집니다.**

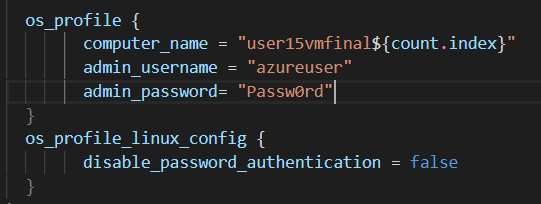


**그리고 Terraform에서는 VM을 생성할 때 다음과 같이 위에서 지정한 key-pair와 동일한 값의 이름을 지정해주셔야 하는데요.**

****

**본 예제의 경우 AWS VM은 위의 과정을 모두 진행하여야 접속이 가능합니다만,**

**Azure의 경우는 따로 키 값을 설정하지 않고, 다음과 같이 패스워드만으로 로그인이 가능하도록 하는 옵션을 넣었습니다. 키 부분 생략하시고 지정하신 ID, Password로 접속하시면 됩니다.**



**Terraform을 활용한 클라우드 구축(개인-심화)**

**이제 Terraform을 이용해 훌륭히 리소스를 만들고. VM에 접속하는 법을 배우셨습니다.**

**다만 위에서 정의한 리소스들의 실제 동작을 위해서는, 위에서 규정한 것들 외의 것들도 구성을 해주어야 합니다.**

**기본 리소스들을 보조하기 위해 따로 정의해주어야 하는 리소스들로 이해하시면 됩니다.**

**Azure의 경우   
- 리소스 그룹(azurerm\_resource\_group) 정의  
- 로드 밸런서에 할당할 Public ip  
- 각 VM 접속에 필요한 Public ip  
- VM에 할당할 nic  
- 로드 밸런서의 상태를 확인하는 lb\_probe  
- 로드 밸런서의 규칙을 정의하는 lb\_rule  
- 로드 밸런서의 백엔드를 정의하는 backend\_address\_pool  
- 위의 백엔드풀에 NIC를 연결하는 network\_interface\_backend\_address\_pool\_association  
- (선택) 손쉽게 VM을 여러 개 만들기 위한 Avset**

**AWS의 경우**

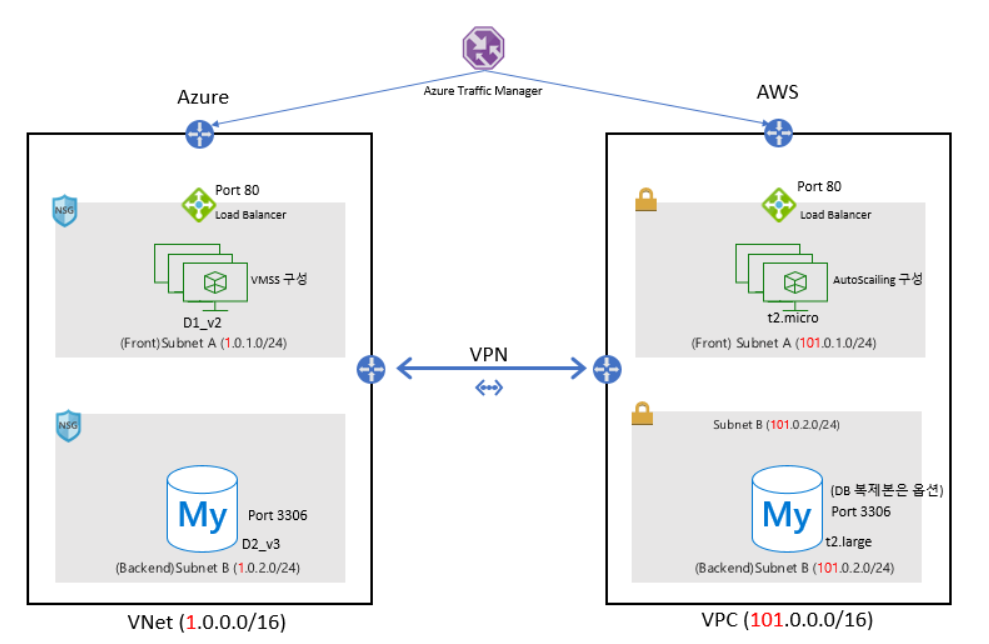
**- 외부 인터넷 개방을 위한 aws\_internet\_gateway  
 - aws\_internet\_gateway와 VPC를 연결하는 Route Table  
 - route\_table과 서브넷을 연결하는** **aws\_route\_table\_association  
 - 각 VM 접속에 필요한 eip  
 - 각 subnet을 위한 Nat table  
 - VPC 네트워크 컨트롤을 위한 aws\_default\_network\_acl  
 - Application Loadbalancer(이하 alb)의 타겟을 정의하는 aws\_alb\_target\_group  
 - alb 타겟 그룹 내의 세부 타겟(VM)을 정의하는 aws\_alb\_target\_group** **\_attachment  
 - 웹으로 어떤 요청이 들어와야 작동하는지를 정의하는 alb의 alb\_listener**

**위를 하나하나 찾아서 구현하고 이해하는 것은 시간이 조금 오래 걸리는데요.**

**문법을 이해하실 수 있도록 과제 코드를 공유 드렸으니. 이를 확인해보시고 하나하나 미리 한번 연습해보시면 도움이 되실 것 같습니다.**

**Terraform을 활용한 클라우드 구축(단체)**

**다음은 단체 과제에 대해서입니다. 기본적으로는 개인 과제와 동일한데. 하부 구조가 더 복잡하고(DB포함) 여러가지 신 기능이 추가된 버전입니다.**



**여기서 추가된 것들은**

**0. 다른 서브넷에 DB서버 추가  
1. Azure VMSS   
2. AWS AutoScaling  
3. 멀티 클라우드 VPN**

**인데요.**

**VM을 직접 생성하는 개인 과제와 달리. VMSS와 AutoScaling 셋을 생성하고, 그 안에 VM으로 사용될 이미지를 넣는 식으로 Terraform을 작성해 주시면 됩니다.**

**관련 내용을 확인해보시고, 개인 과제와 마찬가지로 문법을 확인해보시면 좋을 것 같습니다.**

**\* 저희 차수에서는 실습시 오른쪽 AWS VPC에서는 DB를 만들지 않았습니다.**

**정리**

**지금까지 Terraform을 활용한 AWS/Azure 실습 과정에 대해 알아보았습니다.**

**참고하시라고, 실제로 제가 제출했던 개인/그룹 용 과제 코드도 공유 드립니다.**

**개인용 과제 코드 :**

<https://github.com/leprejin/forcnctest/blob/master/user15-master.zip>

**그룹용 과제 코드 :**

<https://github.com/leprejin/forcnctest/blob/master/group5-master.zip>

**\* 코드 확인 후 Provider.tf 따로 생성하여 사용**

**제 경우에는 실제로는 트래픽 매니저 등이 실습 시에 생각처럼 잘 생성이 안 되어서 포털을 이용해 생성하였는데요.**

**그 외에도 실제 적용시 변수가 있으니 설명 드린 대로 바로 적용하시기 보다 문법만 참고하는 용도로 사용하시는 게 더 좋습니다.**

**마지막으로는 제가 느낀 Tip을 공유 드리고자 합니다.**

**Tip 1 : 실제 시험시 유의점은?**

* **제시해드린 코드와 실제 시험에서 제시하는 구조가 달라질 수 있습니다. 문법은 참고만 하시고, 하나하나 머리에 그리며 직접 만드셔야 코드가 꼬이지 않습니다.**

**Tip 2 : 원하는 대로 코드가 적용되지 않을 때는?**

* **오류 메시지에 답이 있습니다. 대부분은 오타나 클라우드상 중복된 Name 문제입니다.**
* **오류가 계속될 경우 생성되지 않는 부분은 Skip하고 GUI로 수정해주는 것도 방법입니다.**
* **IP가 필요한 로드밸런서, VPN, 트래픽 매니저 등의 리소스는 VM 등의 기본 리소스가 생성된 이후 동작합니다. 먼저 기본 리소스(하부 구조)를 생성하신 후 생성된 정보(IP)를 코드로 입력하여 추가로 Terraform을 실행한다면 더욱 원활한 관리가 가능합니다.**

**Tip 3 : 생성후 VM 웹 서버 세팅에서 유의점은?**

* **실습 시험인 만큼 다 Open해주세요. 특히 RHEL VM은 방화벽 기본이라 꺼주셔야 합니다.**