5.Write IaC using terraform to create EC2 machine on aws or azure or google cloud. (Compulsory to use Input and output variable files)

- Name: Om Deore
- PRN: 22110318
- Div: TY(A)
- RollNo: 321022
- github: All Assignments

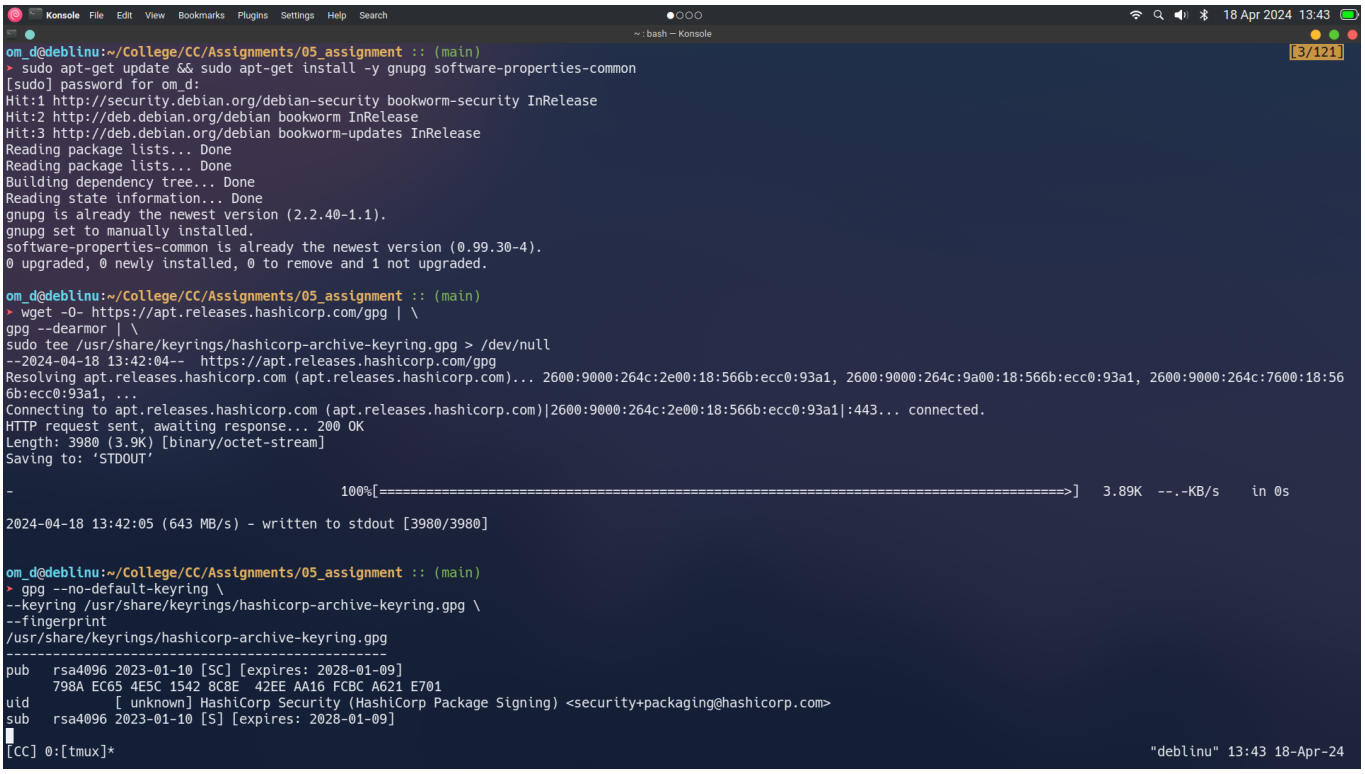
What is Terraform

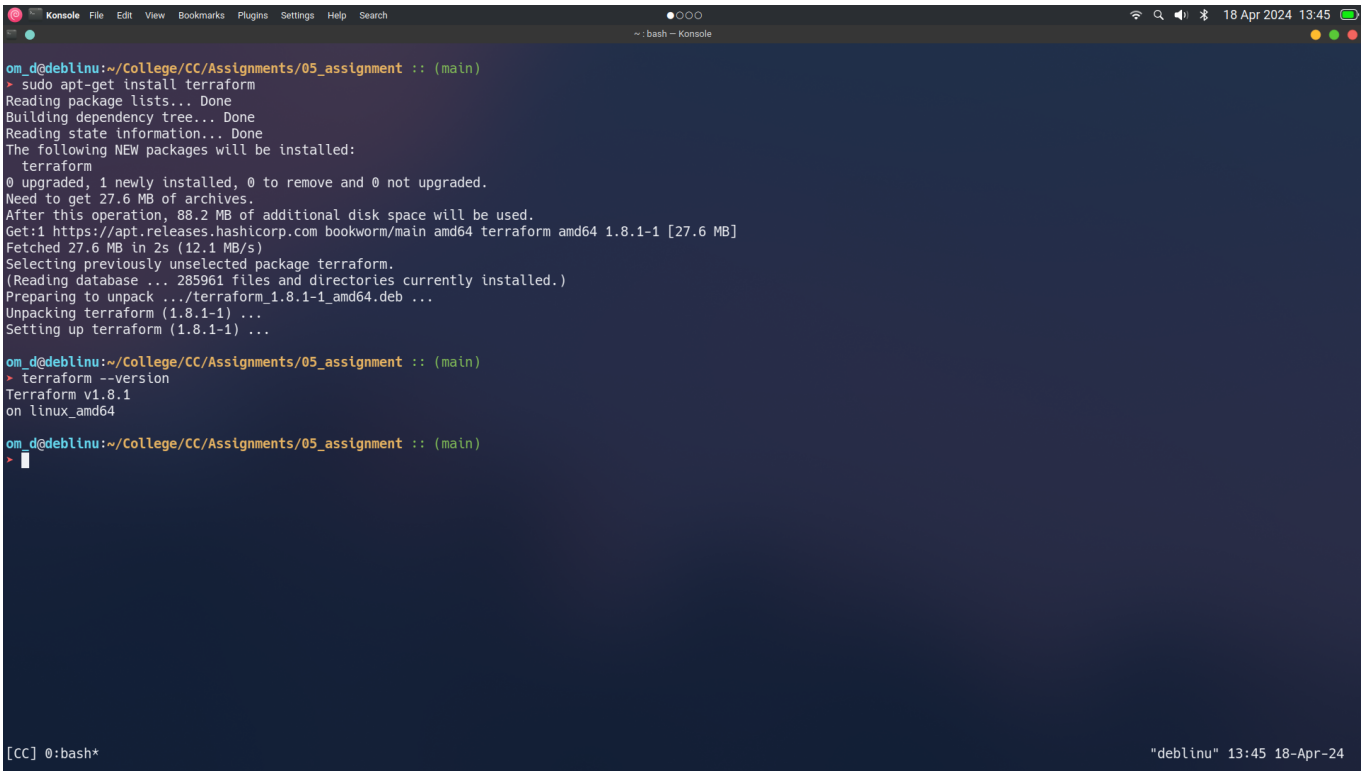
HashiCorp Terraform is an **infrastructure as code** tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

Step-by-step screenshot to install and configure Terraform.

Step1) Installation
To install Terraform on debian linux use following scripts

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-
common
wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update
sudo apt-get install terraform
```

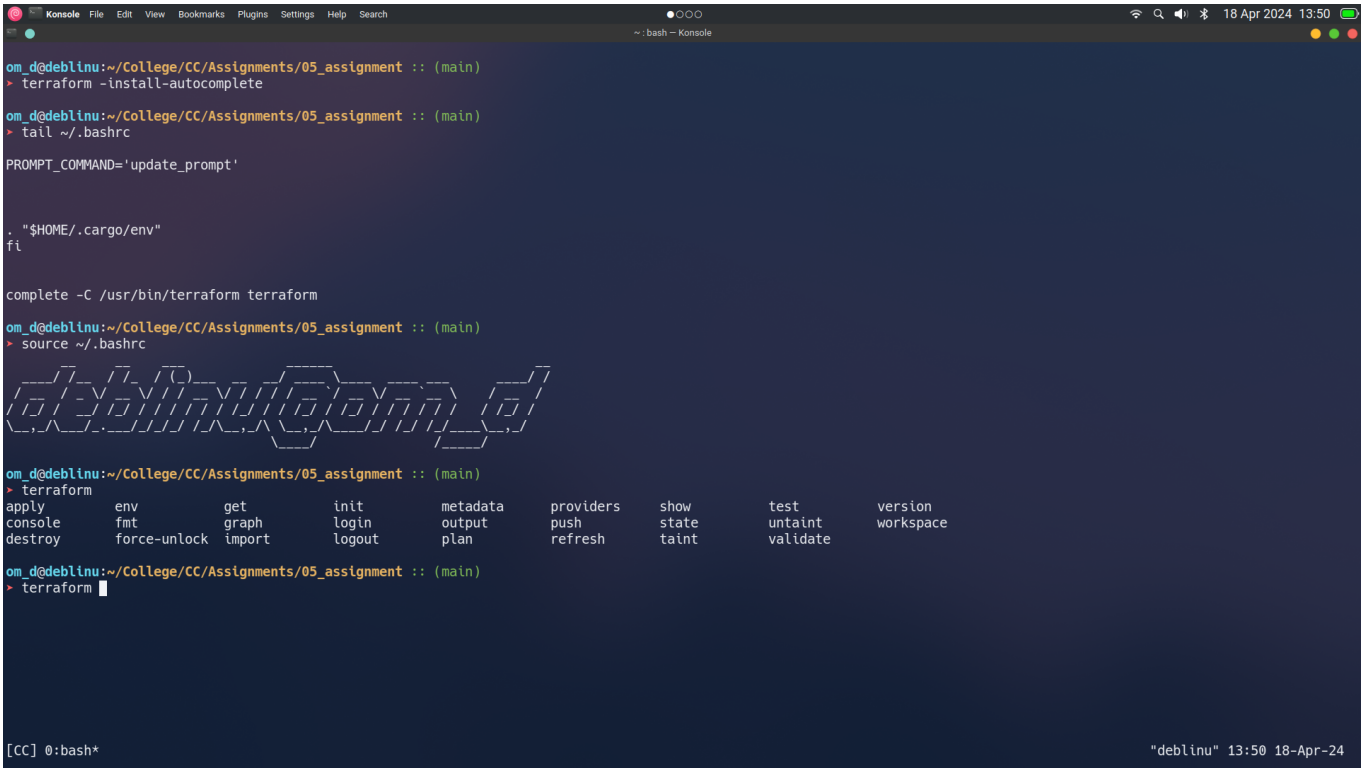




To check if it's installed correctly run

```
$ terraform --version
```

to enable auto complete run following command and source .bashrc to activate it



Here we can see autocomplete being enabled after typing terraform and pressing tab key

Step 2) Install and Configure AWS cli

To install AWS cli run following script

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

To get AWS credentials from your account go to AWS console, click on `profile>Security credentials>Access keys`

Click on `create Access key` to create new access key. It will generate you new "access key" and a "secret access key"

```
export AWS_ACCESS_KEY_ID=<YOUR_ACCESS_KEY>
export AWS_SECRET_ACCESS_KEY_ID=<YOUR_SECRET_ACCESS_KEY>
```

Step 3) Terraform configuration

The set of files used to describe infrastructure in Terraform is known as a Terraform *configuration*. You will write your first configuration to define a single AWS EC2 instance.

Each Terraform configuration must be in its own working directory. Create a directory for your configuration.

in that directory create a `main.tf` file

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region  = "eu-north-1"
}

resource "aws_instance" "app_server" {
  ami           = var.instance_ami_id
  instance_type = var.instance_type

  tags = {
    Name = var.instance_name
  }
}
```

In this file we can see `var` being used. This is for input variables. In same directory create another `variables.tf` file

```
variable "instance_name" {
  description = "Value of the Name tag for the EC2 instance"
  type        = string
  default     = "EC2_using_Terraform"
}

variable "instance_ami_id" {
    description = "Amazon machine image id"
    type = string
    default ="ami-0914547665e6a707c"
}
```

```
variable "instance_type" {
    description = "EC2 instance type"
    type = string
    default = "t3.micro"
}
```

Here we have created few variables a `instance_name`, `instance_ami_id`, `instance_type`
These variables are later used in `main.tf` file. Input variables are useful as they make terraform code modular and increase readability of our code.

To get outputs from terraform we use `outputs.tf` file

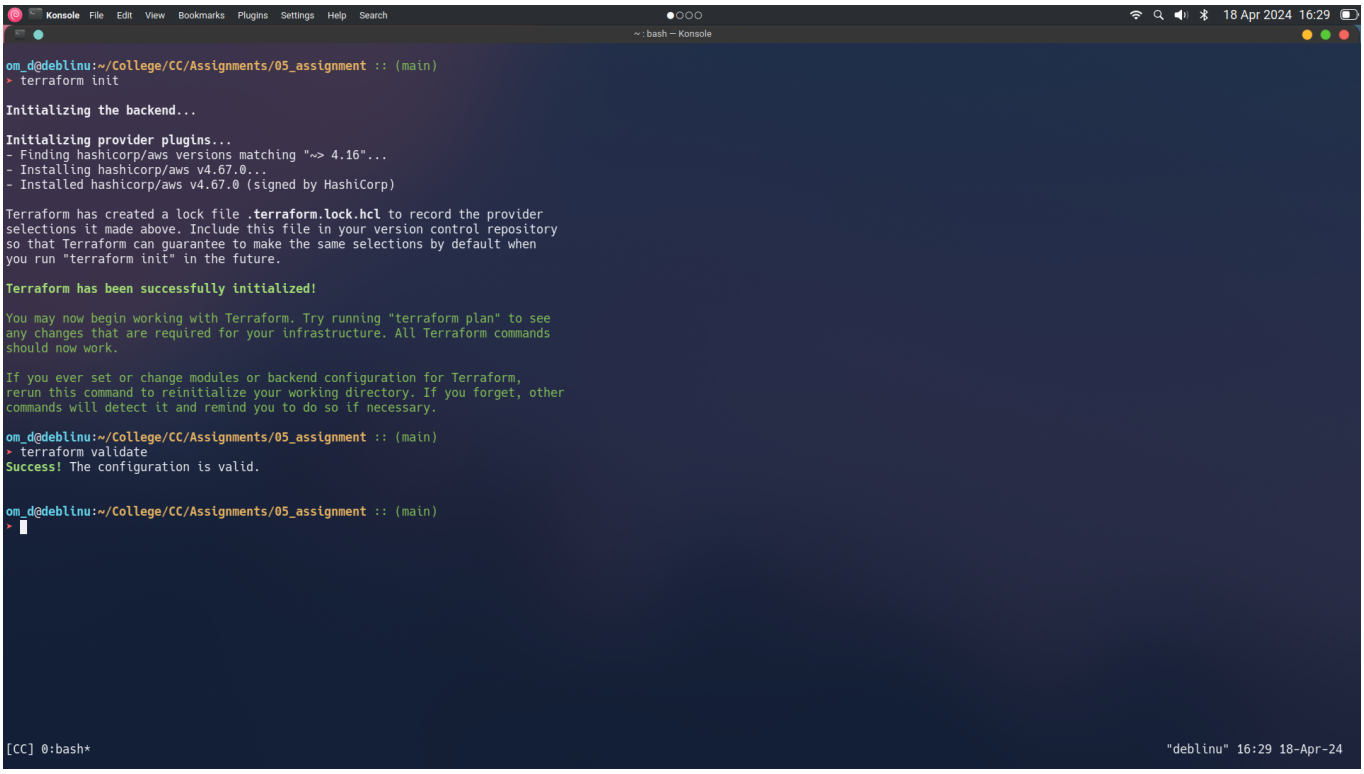
```
output "instance_id" {
  description = "ID of the EC2 instance"
  value       = aws_instance.app_server.id
}

output "instance_public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.app_server.public_ip
}
```

In this demo we create two output variables `instance_id` & `instance_pubilc_ip`
When we apply our terraform configuration we get values of these output variables.

We have our configuration ready and we can launch our EC2 instance from terraform.

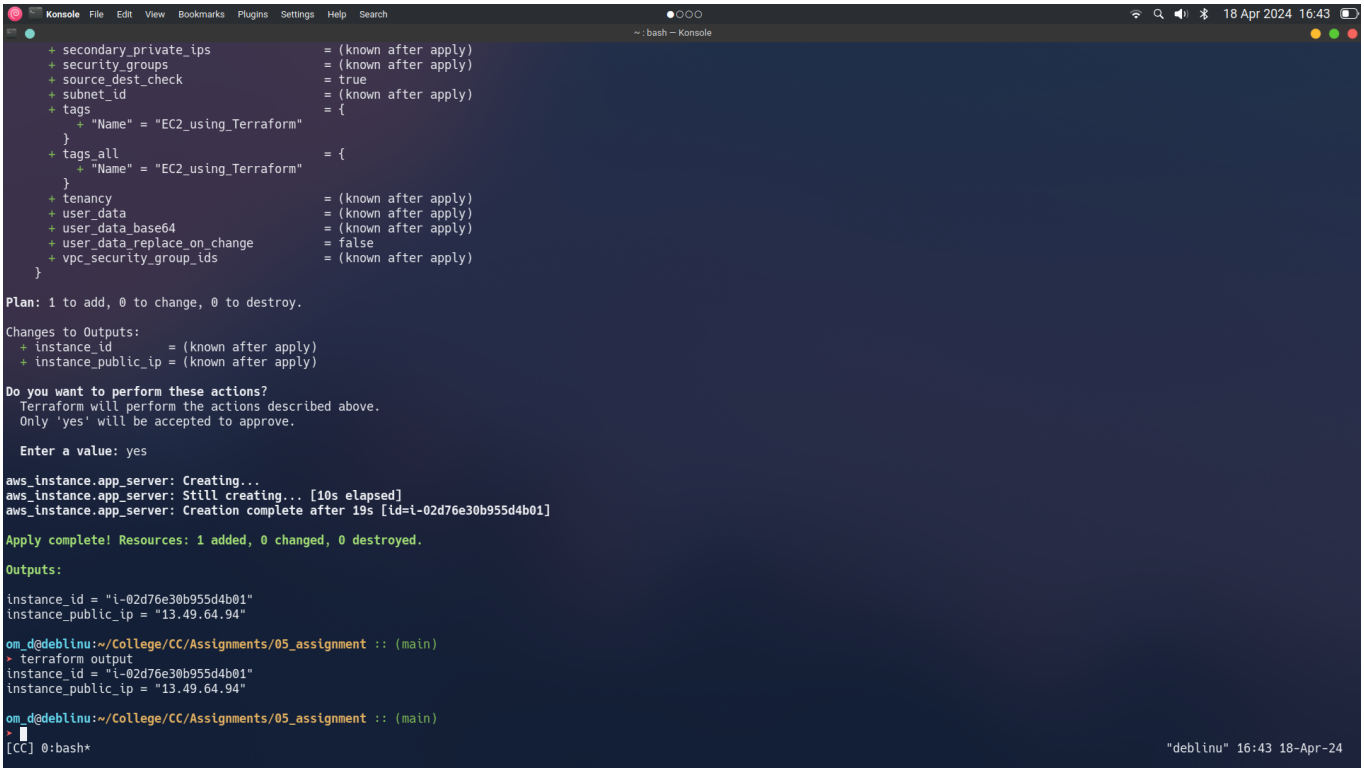
step 4) launch instance using Terraform
We have created all the configurations required to create our EC2 instance using terraform. Now only thing left are few commands

1. `terraform init`
   When you create a new configuration — or check out an existing configuration from version control — you need to initialize the directory with `terraform init`.
   Initializing a configuration directory downloads and installs the providers defined in the configuration, which in this case is the `aws` provider.
   You can also make sure your configuration is syntactically valid and internally consistent by using the `terraform validate` command.



2. `terraform apply`

   To launch instance use this command. Apply command applies any changes we made to existing configuration or launches new infrastructure if it's first time being run. It will display all the changes it will make to infrastructure and prompt you to say yes to reflect these changes.

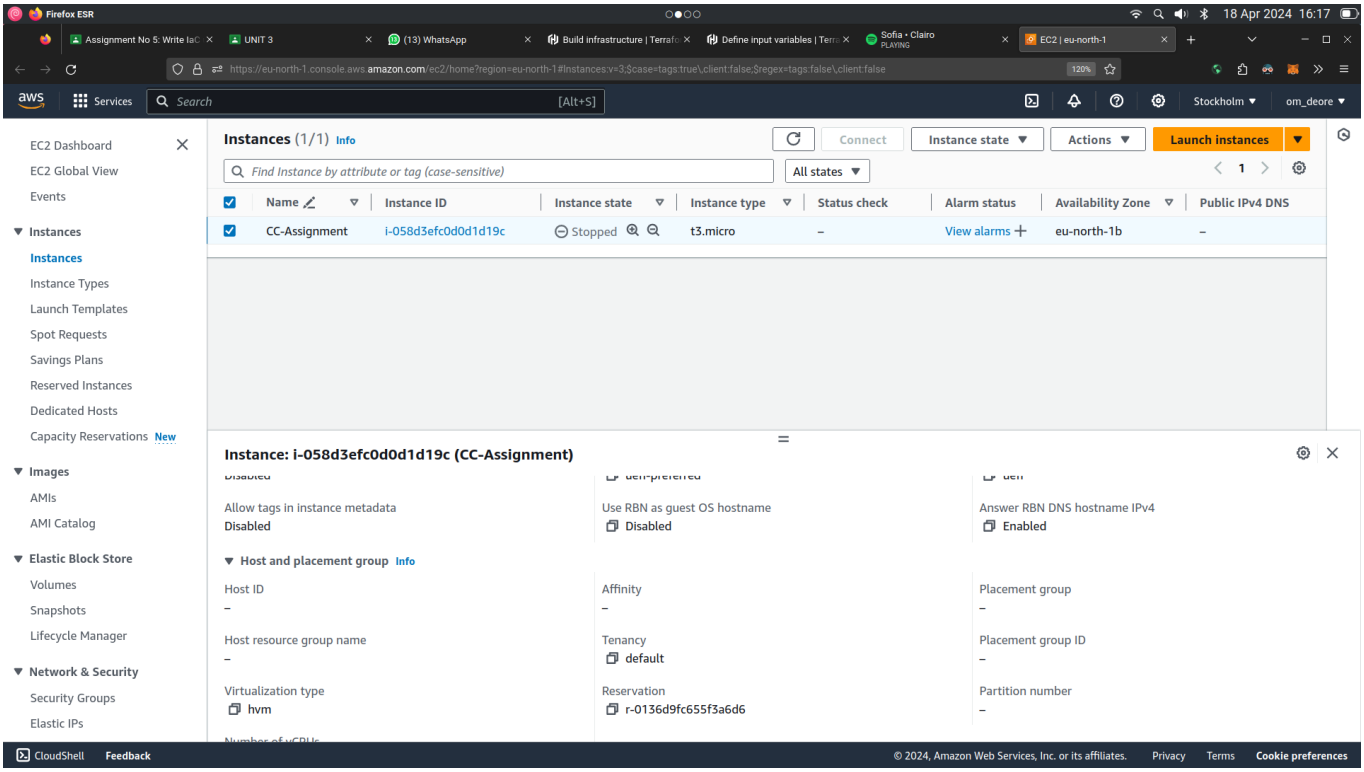


Once this command runs successfully, we can see values to our `output` variables being print. To see these variables again run `terraform output` command

The state of our infrastructure is saved in file called `terraform.tfstate`

We can see our instance being launched

Before apply command



After apply command