Terraform Cheat Sheet

### Basic Commands

- **terraform init**: Initializes a new or existing Terraform configuration.

  - Prepares the working directory for other Terraform commands.

  - Downloads the necessary provider plugins.

- **terraform plan**: Creates an execution plan.

  - Shows what actions Terraform will take to achieve the desired state.

  - Use -out=<plan> to save the plan to a file for later use.

- **terraform apply**: Applies the changes required to reach the desired state of the configuration.

  - Use -auto-approve to skip interactive approval.

- **terraform destroy**: Destroys the Terraform-managed infrastructure.

  - Use -auto-approve to skip interactive approval.

- **terraform validate**: Validates the Terraform files for syntax and logical errors.

- **terraform fmt**: Formats the Terraform files to a canonical style.

- **terraform show**: Displays the state or plan file in a human-readable format.

  - terraform show <statefile>: Display a state file.

  - terraform show <planfile>: Display a plan file.

- **terraform output**: Reads and outputs the variables from the state file.

  - terraform output <variable_name>: Displays the value of a specific output variable.

### Configuration Files

- **main.tf**: Main configuration file where resources are defined.

- **variables.tf**: File to define input variables.

- **outputs.tf**: File to define output values.

- **terraform.tfvars**: File to define the values of the variables.

### Common Blocks

- **Provider Block**: Defines the provider (e.g., AWS, Azure, GCP).

```
provider "aws" {
  region = "us-west-2"
}
```

- **Resource Block**: Defines a resource to be managed.

```
resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

- **Variable Block**: Defines input variables.

```
variable "instance_type" {
  description = "Type of EC2 instance"
  type        = string
  default     = "t2.micro"
}
```

- **Output Block**: Defines output values.

```
output "instance_id" {
  value = aws_instance.example.id
}
```

### State Management

- **terraform state list**: Lists all resources in the state file.
- **terraform state show <resource>**: Shows detailed information about a resource in the state file.
- **terraform state mv <source> <destination>**: Moves a resource from one state to another.
- **terraform state rm <resource>**: Removes a resource from the state file.
- **terraform state pull**: Retrieves the state from its remote location and outputs it to stdout.
- **terraform state push**: Updates remote state from the local state file.

### Workspaces

- **terraform workspace list**: Lists all existing workspaces.
- **terraform workspace show**: Shows the current workspace.
- **terraform workspace new <name>**: Creates a new workspace.
- **terraform workspace select <name>**: Switches to another workspace.
- **terraform workspace delete <name>**: Deletes a workspace.

### Modules

- **Module Block**: Calls a reusable module.

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  version = "2.21.0"

  name = "my-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["us-west-2a", "us-west-2b", "us-west-2c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  tags = {
    Terraform = "true"
    Environment = "dev"
  }
}
```

### Version Control

- **.terraform.lock.hcl**: File to lock the provider versions.

- **terraform version**: Shows the Terraform version.

- **terraform providers**: Lists all the providers used in the configuration.

### Environment Variables

- **TF_VAR_name**: Sets a variable (e.g., export TF_VAR_region=us-west-2).

- **TF_LOG**: Sets the log level (e.g., export TF_LOG=DEBUG).

- **TF_CLI_ARGS**: Appends extra CLI arguments to Terraform commands.

### Terraform Cloud and Backend

- **Backend Configuration**: Defines where the state file is stored.

```
terraform {
  backend "s3" {
    bucket = "my-terraform-state"
    key    = "path/to/my/key"
    region = "us-west-2"
  }
}
```

- **Terraform Cloud**: Using Terraform Cloud as the backend.

```
terraform {
  backend "remote" {
    organization = "my-org"

    workspaces {
      name = "my-workspace"
    }
  }
}
```