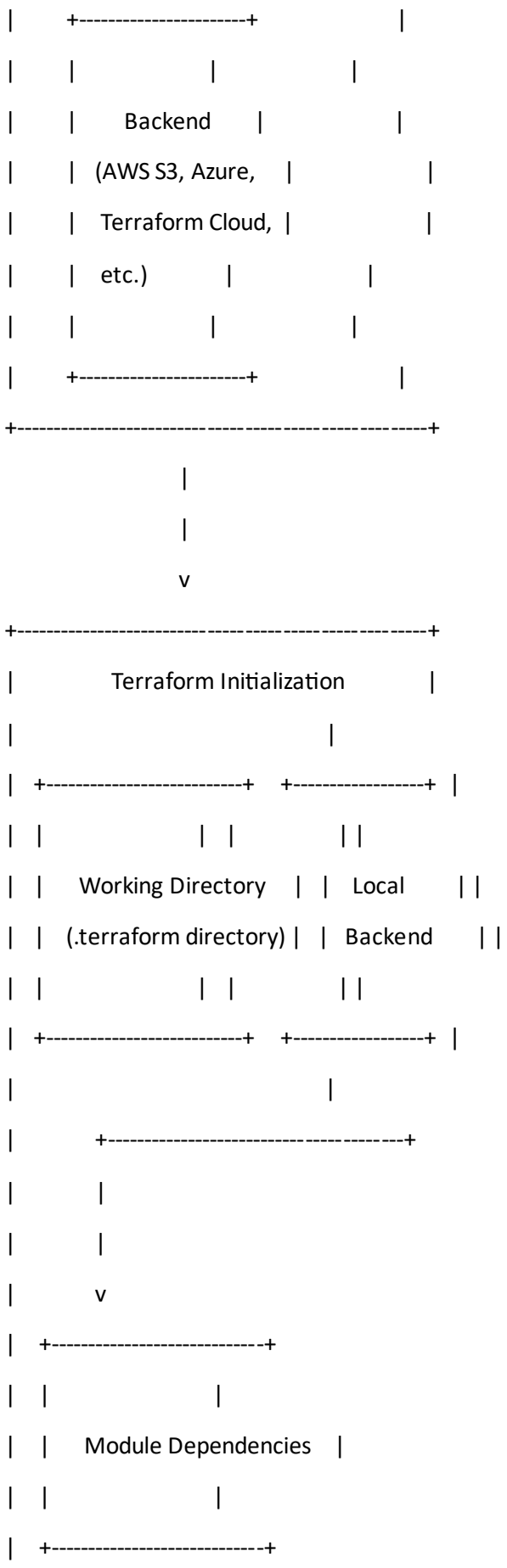
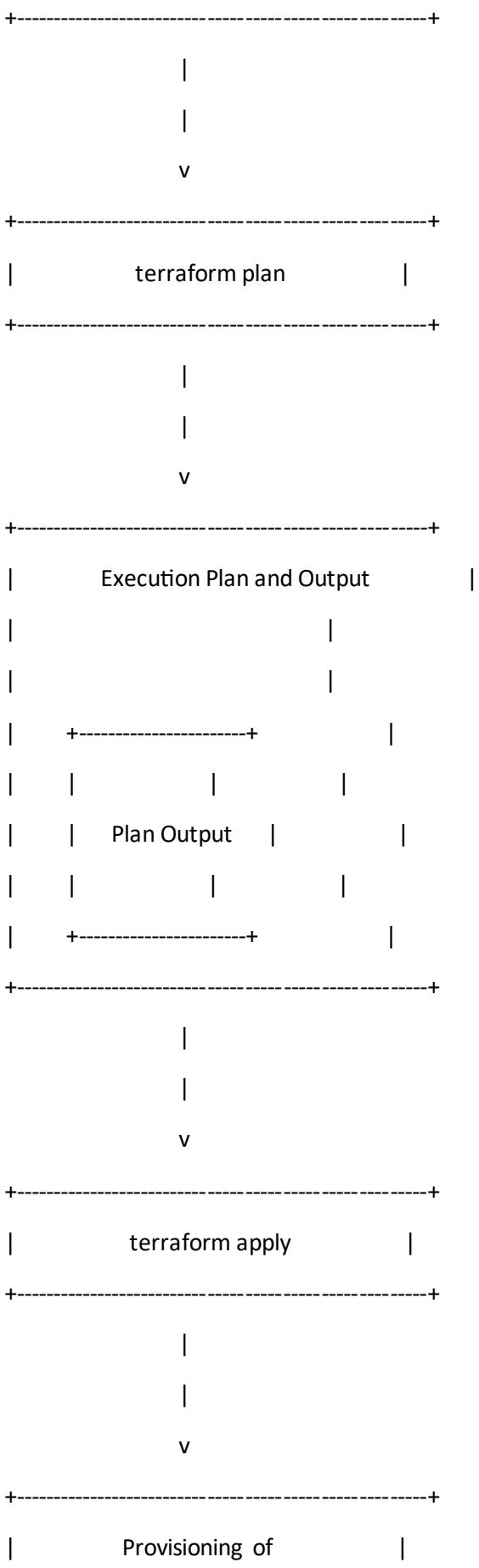
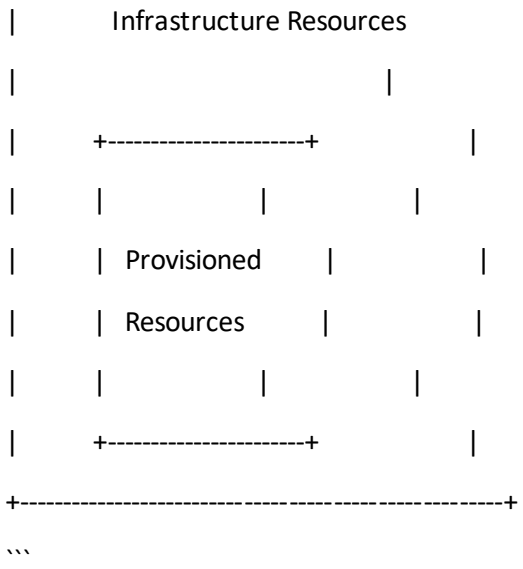
```
+-----------------------------------------------------+
|                  terraform init                     |
+-----------------------------------------------------+
                        |
                        |
                        v
+-----------------------------------------------------+
|               Configuration Files                   |
|            (main.tf, variables.tf, etc.)            |
+-----------------------------------------------------+
                        |
                        |
                        v
+-----------------------------------------------------+
|             Provider Plugin Installation            |
|                                     |
|         +----------------------+                    |
|         |                      |                    |
|         |      Providers       |                    |
|         |   (AWS, Azure, etc.) |                    |
|         |                      |                    |
|         +----------------------+                    |
+-----------------------------------------------------+
                        |
                        |
                        v
+-----------------------------------------------------+
|              Backend Initialization                 |
|                                     |
```


Backend
(AWS S3, Azure,
Terraform Cloud,
etc.)
Terraform Initialization
Working Directory
(.terraform directory)
Local
Backend
Module Dependencies


terraform plan
Execution Plan and Output
Plan Output
terraform apply
Provisioning of




```
```

Let's understand the workflow represented in the block diagram:

1. **`terraform init`**: This command initializes the Terraform working directory, downloads provider plugins, sets up the backend connection, and initializes the local backend and module dependencies.

2. **Configuration Files**: These files define the desired state of your infrastructure and specify the resources, their configurations, and any variables.

3. **Provider Plugin Installation**: Terraform identifies the required provider plugins specified in the configuration files and downloads them. These plugins allow Terraform to interact with the infrastructure providers.

4. **Backend Initialization**: If a backend is configured, Terraform initializes the backend connection. The backend is responsible for storing the state and managing remote operations like locking and state retrieval.

5. **Terraform Initialization**: Terraform sets up the working directory, initializes the local backend (if used),

and resolves and downloads any module dependencies.

6. **Module Dependencies**: If your configuration uses modules, Terraform resolves and downloads the required module code from the Terraform Registry or a custom module source. The module code is stored in the `.terraform/modules` directory.

7. **`terraform plan`**: This command generates an execution plan based on the current state and desired state defined in the configuration files. It compares the two states, identifies the changes required to reach the desired state, and provides a detailed plan of the proposed changes.

8. **Execution Plan and Output**: The plan output shows the resources to be created, modified, or destroyed. It provides information about the actions Terraform will take during the next `apply` operation.

9. **`terraform apply`**: This command applies the changes defined in the plan. It provisions or modifies the infrastructure resources according to the desired state defined in the configuration files.

10. **Provisioning of Infrastructure Resources**: Terraform interacts with the infrastructure providers, creating, modifying, or destroying resources as necessary to achieve the desired state.

The `terraform init`, `terraform plan`, and `terraform apply` commands form a typical workflow in Terraform, allowing you to initialize the project, plan and preview the changes, and finally apply those changes to provision or modify your infrastructure resources.