Install Terraform

1) Download Terraform package
https://www.terraform.io/downloads.html

2) Add Terraform executable to the Path

3) Verify Terraform install
$ C:\opt\terraform_0.10.7_windows_amd64>terraform.exe

Output:

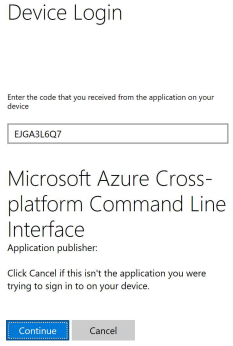
```
C:\opt\terraform_0.10.7_windows_amd64>terraform.exe
Usage: terraform [--version] [--help] <command> [args]
The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version
    workspace          Workspace management

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
Usage: terraform [--version] [--help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.
```

4) Set up Terraform access to Azure
To enable Terraform to provision resources into Azure, you need to create two entities in Azure Active Directory (Azure AD): an Azure AD application and an Azure AD service principal.

Azure env setup: provider.azurerm
Run `az login` to obtain Azure CLI Auth Tokens
$ az login

Output:
C:\MyWork\TE\Clients\Amperity\TestLabs\Terraform\azurerm_container_service>az login
To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code EJGA3L6Q7 to authenticate.

4.1) Go to browser and navigate to: https://aka.ms/devicelogin
4.2) Azure authentication with Device Login code: EJGA3L6Q7



4.3) Click <Continue> > to select azure account to login
4.4) Azure CLI - Azure authenticated

4.5) Go back to CLI - Completed authentication with Azure


```
[
  {
    "cloudName": "AzureCloud",
    "id": "c27{...}c1c",
    "isDefault": true,
    "name": "Visual Studio Enterprise",
    "state": "Enabled",
    "tenantId": "bf5{...}9d3",
    "user": {
      "name": "rsliang@yahoo.com",
      "type": "user"
    }
  }
]
```


5) Query account for subscription ID and tenant ID:
$ az account show --query "{subscriptionId:id, tenantId:tenantId}"
Output:
C:\MyWork\TE\Clients\Amperity\TestLabs\Terraform\azurerm_container_service>az account

show --query "{subscriptionId:id, tenantId:tenantId}"
{
 "subscriptionId": "c27{...}c1c",
 "tenantId": "bf5{...}9d3"
}

6) Set the subscription for the session
$ az account set --subscription="${SUBSCRIPTION_ID}"
Output:
C:\opt\terraform_0.10.7_windows_amd64>az account set --subscription="c27{...}c1c"

7) Create separate credential for Terraform
$ az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/${SUBSCRIPTION_ID}"
Output:
C:\MyWork\TE\Clients\Amperity\TestLabs\Terraform\azurerm_container_service>az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/c27{...}c1c"
{
 "appId": "{...}b82",
 "displayName": "azure-cli-{...}",
 "name": "http://azure-cli-{...}",
 "password": "b65{...}be1",
 "tenant": "bf5{...}9d3"
}

8) Set environment variables (optional)
After you create and configure an Azure AD service principal, you need to let Terraform know the tenant ID, subscription ID, client ID, and client secret to use. You can do it by embedding those values in your Terraform scripts, as described in Create basic infrastructure by using Terraform. Alternately, you can set the following environment variables (and thus avoid accidentally checking in or sharing your credentials):+
ARM_SUBSCRIPTION_ID
ARM_CLIENT_ID
ARM_CLIENT_SECRET
ARM_TENANT_ID

Sample shell script:
#!/bin/sh
echo "Setting environment variables for Terraform"
export ARM_SUBSCRIPTION_ID=your_subscription_id
export ARM_CLIENT_ID=your_appId
export ARM_CLIENT_SECRET=your_password
export ARM_TENANT_ID=your_tenant_id

Build ACS with Meso DCOS container orchestrator using Terraform:

Creates an Azure Container Service Instance

Note: All arguments including the client secret will be stored in the raw state as plain-text. Read more about sensitive data in state.

Terraform template: azurerm_container_service

azurerm_container_service.tf:

```
resource "azurerm_resource_group" "test" {
  name     = "demo-acs-dcos-tf-eastus-rg"
  location = "East US"
}

resource "azurerm_container_service" "test" {
  name                = "acctestcontservice1"
  location            = "${azurerm_resource_group.test.location}"
  resource_group_name    = "${azurerm_resource_group.test.name}"
  orchestration_platform = "DCOS"

  master_profile {
    count     = 1
    dns_prefix = "acctestmaster1-{...}
  }

  linux_profile {
    admin_username = "acctestuser1"

    ssh_key {
      key_data = "ssh-rsa AAA{...}1CR terraform@demo.tld"
    }
  }

  agent_pool_profile {
    name      = "default"
    count     = 1
    dns_prefix = "acctestagent1-{...}"
    vm_size    = "Standard_A0"
  }

  diagnostics_profile {
    enabled = false
  }

  tags {
    Environment = "Demo"
  }
}
```

1) Initialize Terraform.

$ terraform init

Output:

C:\MyWork\TE\Clients\Amperity\TestLabs\Terraform\azurerm_container_service>terraform init

```
Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "azurerm" (0.3.0)...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.azurerm: version = "~> 0.3"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2) Terraform review and validate the template.

This step compares the requested resources to the state information saved by Terraform and then outputs the planned execution. Resources are not created in Azure.

$ terraform plan

Output:

```
C:\MyWork\TE\Clients\Amperity\TestLabs\Terraform\azurerm_container_service>terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

azurerm_resource_group.test: Refreshing state... (ID: /subscriptions/c27{...}e5c-...ourceGroups/demo-acs-dcos-tf-eastus-rg)

------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + azurerm_container_service.test
      id:                                       <computed>
      agent_pool_profile.#:                     "1"
      agent_pool_profile.2827755561.count:      "1"
      agent_pool_profile.2827755561.dns_prefix: "acctestagent1-{...}
      agent_pool_profile.2827755561.fqdn:       <computed>
      agent_pool_profile.2827755561.name:       "default"
      agent_pool_profile.2827755561.vm_size:    "Standard_A0"
      diagnostics_profile.#:                    "1"
      diagnostics_profile.734881840.enabled:    "false"
      diagnostics_profile.734881840.storage_uri: <computed>
      linux_profile.#:                          "1"
      linux_profile.2765581951.admin_username:  "acctestuser1"
      linux_profile.2765581951.ssh_key.#:       "1"
```

```
    linux_profile.2765581951.ssh_key.1472416176.key_data: "ssh-rsa AAA{...}1CR terraform@demo.tld"
    location:                                 "eastus"
    master_profile.#:                             "1"
    master_profile.3882221260.count:              "1"
    master_profile.3882221260.dns_prefix:           "acctestmaster1-{...}"
    master_profile.3882221260.fqdn:               <computed>
    name:                                     "acctestcontservice1"
    orchestration_platform:                       "DCOS"
    resource_group_name:                        "demo-acs-dcos-tf-eastus-rg"
    tags.%:                                   "1"
    tags.Environment:                             "Demo"


Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

3) build the infrastructure in Azure, apply the template in Terraform

$ terraform apply

Output:

```
C:\MyWork\TE\Clients\Amperity\TestLabs\Terraform\azurerm_container_service>terraform apply
azurerm_resource_group.test: Creating...
  location: "" => "eastus"
  name:     "" => "demo-acs-dcos-tf-eastus-rg"
  tags.%:   "" => "<computed>"
azurerm_resource_group.test: Creation complete after 1s (ID: /subscriptions/c27{...}e5c-...ourceGroups/demo-acs-dcos-tf-eastus-rg)
azurerm_container_service.test: Creating...
  agent_pool_profile.#:                         "" => "1"
  agent_pool_profile.2827755561.count:          "" => "1"
  agent_pool_profile.2827755561.dns_prefix:       "" => "acctestagent1-{...}"
  agent_pool_profile.2827755561.fqdn:           "" => "<computed>"
  agent_pool_profile.2827755561.name:           "" => "default"
  agent_pool_profile.2827755561.vm_size:          "" => "Standard_A0"
  diagnostics_profile.#:                        "" => "1"
  diagnostics_profile.734881840.enabled:        "" => "false"
  diagnostics_profile.734881840.storage_uri:      "" => "<computed>"
  linux_profile.#:                              "" => "1"
  linux_profile.2765581951.admin_username:        "" => "acctestuser1"
  linux_profile.2765581951.ssh_key.#:           "" => "1"
  linux_profile.2765581951.ssh_key.1472416176.key_data: "" => "ssh-rsa AAA{...}1CR terraform@demo.tld"
  location:                                     "" => "eastus"
  master_profile.#:                             "" => "1"
  master_profile.3882221260.count:              "" => "1"
  master_profile.3882221260.dns_prefix:           "" => "acctestmaster1-{...}"
  master_profile.3882221260.fqdn:               "" => "<computed>"
  name:                                         "" => "acctestcontservice1"
  orchestration_platform:                         "" => "DCOS"
  resource_group_name:                          "" => "demo-acs-dcos-tf-eastus-rg"
  tags.%:                                       "" => "1"
  tags.Environment:                               "" => "Demo"
azurerm_container_service.test: Still creating... (10s elapsed)
azurerm_container_service.test: Still creating... (20s elapsed)
...
azurerm_container_service.test: Still creating... (7m0s elapsed)
azurerm_container_service.test: Creation complete after 7m8s (ID: /subscriptions/c27{...}
e5c-.../containerServices/acctestcontservice1)
```

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.