Building Cloud Infrastructure using Terraform

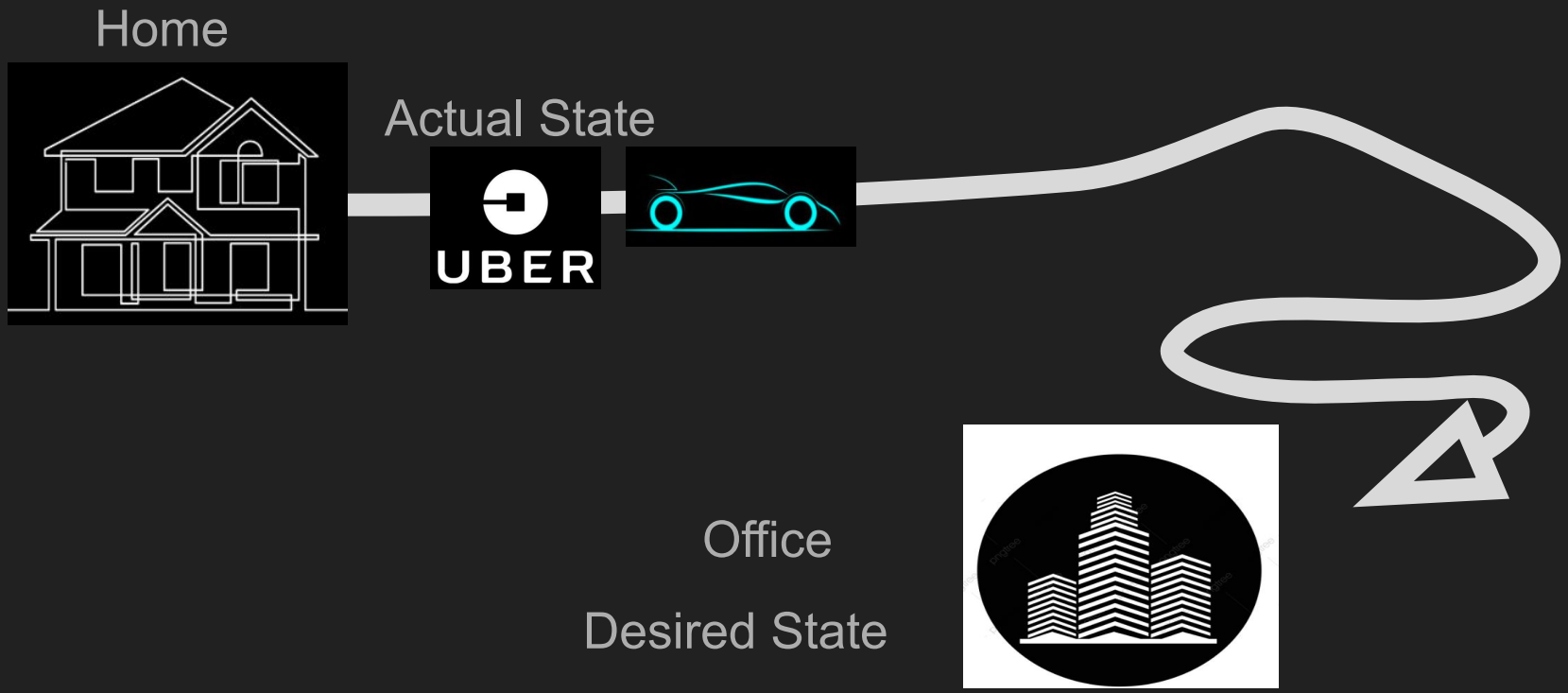
Content

- What is terraform
- Terraform Workflow in myview
- Actual Terraform Workflow
- Which is best ?
- Terraform Features
- Terraform & Docker installation
- Lab & Main.tf
- Questions

What is Terraform ?

- Terraform is a tool for building infrastructure.
- Available as open-source or enterprise software.
- Enterprise provides advanced collaboration and governance.
- Allows simple version control
- Supports many popular service providers such as
  - GCP
  - AWS
  - OpenStack
  - Azure
  - Docker
  - Kubernetes etc..

Terraform Workflow in my view

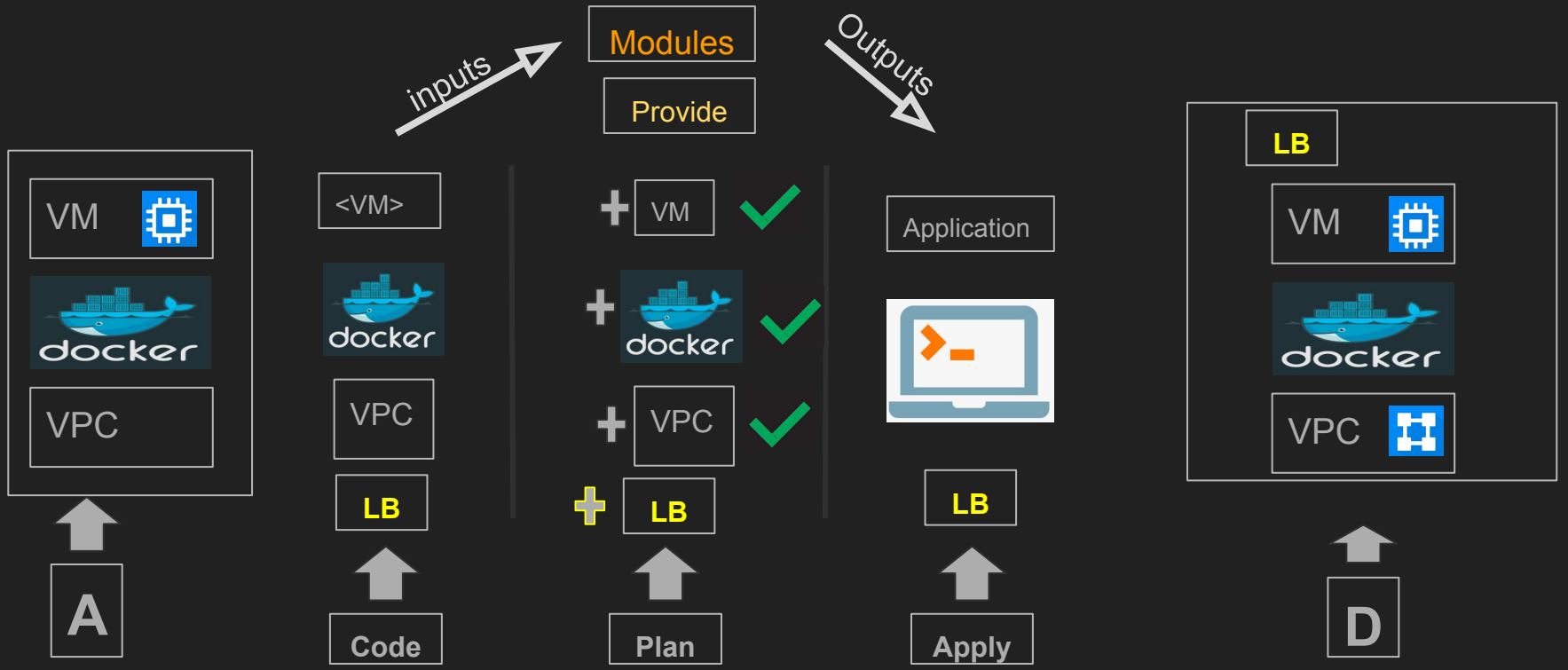


Terraform workflow

Declarative

Plugins

Devops Best Practice



Which is best?

- Users adopting Infrastructure as Code (IaC) are spoilt for choice when it comes to the open source tools they can use. In some shape or form, Chef, Ansible, Puppet, SaltStack, Terraform, and more are all tools that can be used as part of a DevOps toolchain to implement provisioning of app environments and infrastructure.

	Chef	Puppet	Ansible	SaltStack	Terraform
Cloud		All	All	All	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable
Language	Procedural	Declarative	Procedural	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client only	Client only	Client only
Orchestration					
Lifecycle (state) management	No	No	No	No	Yes
VM provisioning	Partial	Partial	Partial	Partial	Yes
Networking	Partial	Partial	Partial	Partial	Yes
Storage Management	Partial	Partial	Partial	Partial	Yes
Configuration					
Packaging	Yes	Yes	Yes	Yes	Partial [1]
Templating	Yes	Yes	Yes	Yes	Partial [1]
Service provisioning	Yes	Yes	Yes	Yes	Yes

[1] Using CloudInit

Terraform Features

Primary Terraform Features.

- Infrastructure as Code (IaC)
  - Idempotent
  - High-Level Syntax
  - Easily reusable - In this git repo you can find some example community modules: https://github.com/terraform-community-modules
- Execution Plans
  - Show the intent of the deploy
  - Can help ensure everything in the development is international
- Resource graph
  - Illustrates all changes and dependencies

Terraform & Docker installation

- Installing Terraform is pretty easy. You can find the download page of the latest version here: https://www.terraform.io/downloads.html.
- `$ sudo curl -O https://releases.hashicorp.com/terraform/0.11.13/terraform_0.11.13_linux_amd64.zip`
- `$ sudo yum install -y unzip`
- `$ sudo unzip terraform_0.11.13_linux_amd64.zip -d /usr/local/bin/`
- Test the Terraform installation:
- `$ terraform version`

Docker Installation

- Update the Operating System
- $ sudo yum update -y
- Add the Docker repository
- $ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
- Install Docker CE:
- $ dnf list docker-ce
- $ dnf install docker-ce --nobest -y
- Start Docker and enable it:
- $ systemctl start docker
- $ systemctl enable docker

Quick Demo to deploy Docker container

- Create a VM in GCP
- Install Docker
- Install Terraform
- Deploy Docker image and run it using Terraform.

.tf File (Main.tf)

Resource Block

Resource Type/attribute coming from the provider

Image that we are going to pull

https://www.terraform.io/docs/providers

Provides

- Terraform is used to create, manage, and update infrastructure resources such as physical machines, VMs, network switches, containers, and more. Almost any infrastructure type can be represented as a resource in Terraform.

- A provider is responsible for understanding API interactions and exposing resources. Providers generally are an IaaS (e.g. Alibaba Cloud, AWS, GCP, Microsoft Azure, OpenStack), PaaS (e.g. Heroku), or SaaS services (e.g. Terraform Cloud, DNSimple, Cloudflare).

Resources

- The most important thing you'll configure with Terraform are resources. Resources are a component of your infrastructure. It might be some low level component such as a physical server, virtual machine, or container. Or it can be a higher level component such as an email provider, DNS record, or database provider.
- The resource block creates a resource of the given TYPE (first parameter) and NAME (second parameter). The combination of the type and name must be unique.

Variables

- Input variables serve as parameters for a Terraform module, allowing aspects of the module to be customized without altering the module's own source code, and allowing modules to be shared between different configurations.
- When you declare variables in the root module of your configuration, you can set their values using CLI options and environment variables.

```
[root@terraform-1 adv]# tree /root/terraform/adv
/root/terraform/adv
├── main.tf
├── Terraform.json
├── terraform.tfstate
├── terraform.tfstate.backup
└── variables.tf

0 directories, 5 files
```

Demo

Create below resources using Terraform

- Create GCP Instance from Terraform.
- Install Docker on GCP instance
- Deploy docker httpd image.
- Deploy basic html file.
- Create firewall rule and enable protocol/port tcp:80.
- Attach Network tag to newly created VM to access httpd on 80.

Terraform State

- Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.
- This state is stored by default in a local file named "terraform.tfstate", but it can also be stored remotely, which works better in a team environment.
- Terraform uses this local state to create plans and make changes to your infrastructure. Prior to any operation, Terraform does a refresh to update the state with the real infrastructure.

Thanks