Terraform driver for Docker Machine

At its heart, Terraform is a way of describing infrastructure.

The idea:

At its heart, Terraform is a way of describing infrastructure.

The idea is that, over time, the infrastructure's configuration converges with its description.

At its heart, Terraform is a way of describing infrastructure.

So the general expectation is Terraform has an ongoing relationship with the infrastructure.

At its heart, Terraform is a way of describing infrastructure.

But this is only one way of using Terraform.

Docker Machine was originally designed to lower the barrier to entry for people wanting to try out Docker.

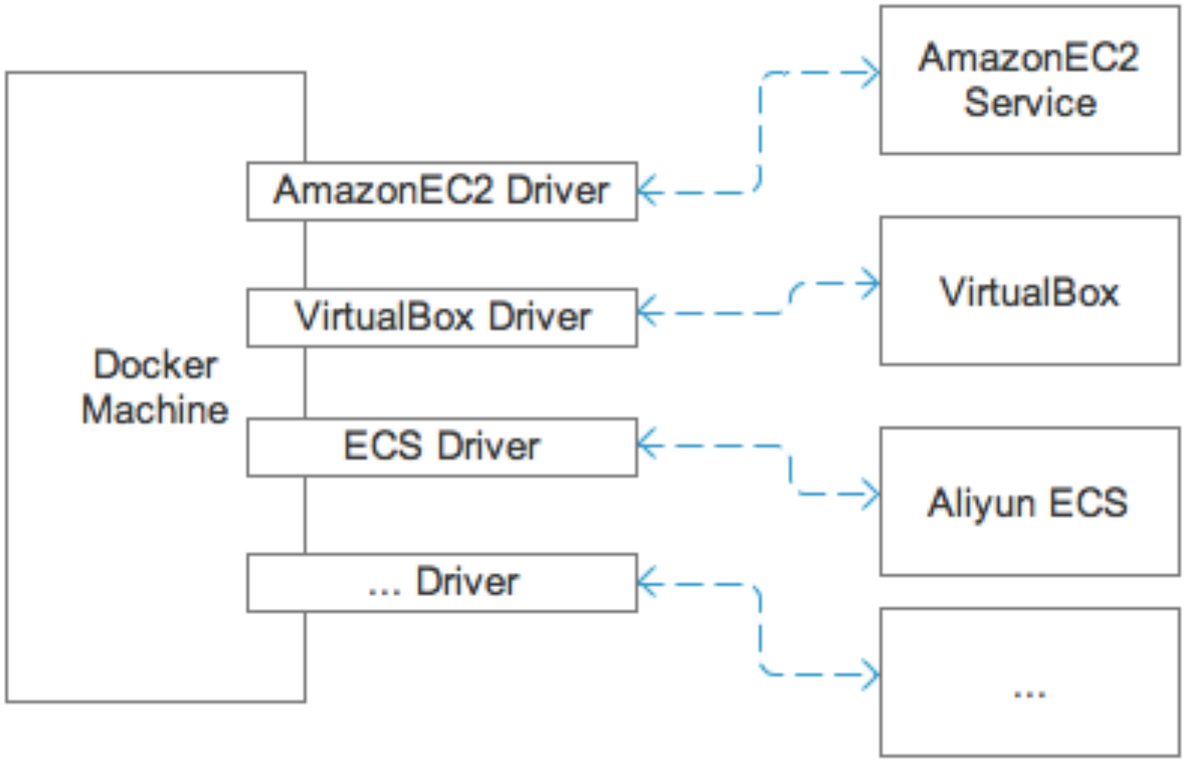
Docker Machine was originally designed to lower the barrier to entry for people wanting to try out Docker.

If a developer or engineer wants to try it out on their laptop, all they need to do is run `docker-machine create` and Docker Machine takes care of the rest.

Docker Machine was originally designed to lower the barrier to entry for people wanting to try out Docker.

Over time, additional drivers have been built to enable deployment of Docker onto many different clouds.

Over time, additional drivers have been built to enable deployment of Docker onto many different clouds.



Over time, additional drivers have been built to enable deployment of Docker onto many different clouds.

So docker Machine is mostly about one-off deployments (and cleaning up if / when they are no-longer needed).

Over time, additional drivers have been built to enable deployment of Docker onto many different clouds.

It is an abstraction over the virtualisation used to deploy Docker hosts.

But ask any 3 cloud providers how to deploy a virtual machine, and you'll get 4 different answers.

But ask any 3 cloud providers how to deploy a virtual machine, and you'll get 4 different answers.

Every cloud is different, both in presentation (UI / API) and implementation.

The dream:

"deploying to the cloud means you can focus on *what* you want rather than *how* it's done"

The dream:

"deploying to the cloud means you can focus on *what* you want rather than *how* it's done"

The reality:

"by all means, focus on this thing you want, but your provider has 5 different ways of accomplishing it, each of which has its own advantages and disadvantages"

If you've seen the Docker Machine driver for AWS, you'll understand why this can be painful:

If you've seen the Docker Machine driver for AWS, you'll understand why this can be painful:

```
Options:

   --amazonec2-access-key
        AWS Access Key [$AWS_ACCESS_KEY_ID]
   --amazonec2-ami
        AWS machine image [$AWS_AMI]
   --amazonec2-block-duration-minutes "0"
        AWS spot instance duration in minutes (60, 120, 180, 240, 300, or 360)
   --amazonec2-device-name "/dev/sda1"
        AWS root device name [$AWS_DEVICE_NAME]
   --amazonec2-endpoint
        Optional endpoint URL (hostname only or fully qualified URI) [$AWS_ENDPOINT]
   --amazonec2-iam-instance-profile
        AWS IAM Instance Profile [$AWS_INSTANCE_PROFILE]
   --amazonec2-insecure-transport
        Disable SSL when sending requests [$AWS_INSECURE_TRANSPORT]
   --amazonec2-instance-type "t2.micro"
        AWS instance type [$AWS_INSTANCE_TYPE]
   --amazonec2-keypair-name
        AWS keypair to use; requires --amazonec2-ssh-keypath [$AWS_KEYPAIR_NAME]
   --amazonec2-monitoring
        Set this flag to enable CloudWatch monitoring
   --amazonec2-open-port [--amazonec2-open-port option --amazonec2-open-port option]
        Make the specified port number accessible from the Internet
   --amazonec2-private-address-only
        Only use a private IP address
   --amazonec2-region "us-east-1"
        AWS region [$AWS_DEFAULT_REGION]
   --amazonec2-request-spot-instance
        Set this flag to request spot instance
   --amazonec2-retries "5"
        Set retry count for recoverable failures (use -1 to disable)
   --amazonec2-root-size "16"
        AWS root disk size (in GB) [$AWS_ROOT_SIZE]
   --amazonec2-secret-key
        AWS Secret Key [$AWS_SECRET_ACCESS_KEY]
   --amazonec2-security-group [--amazonec2-security-group option --amazonec2-security-group option]
     AWS VPC security group [$AWS_SECURITY_GROUP]
   --amazonec2-session-token
        AWS Session Token [$AWS_SESSION_TOKEN]
   --amazonec2-spot-price "0.50"
        AWS spot instance bid price (in dollar)
   --amazonec2-ssh-keypath
        SSH Key for Instance [$AWS_SSH_KEYPATH]
   --amazonec2-ssh-user "ubuntu"
```

```
   --amazonec2-subnet-id
        AWS VPC subnet id [$AWS_SUBNET_ID]
   --amazonec2-tags
        AWS Tags (e.g. key1,value1,key2,value2) [$AWS_TAGS]
   --amazonec2-use-ebs-optimized-instance
        Create an EBS optimized instance
   --amazonec2-use-private-address
        Force the usage of private IP address
   --amazonec2-userdata
        path to file with cloud-init user data [$AWS_USERDATA]
   --amazonec2-volume-type "gp2"
        Amazon EBS volume type [$AWS_VOLUME_TYPE]
   --amazonec2-vpc-id
        AWS VPC id [$AWS_VPC_ID]
   --amazonec2-zone "a"
        AWS zone for instance (i.e. a,b,c,d,e) [$AWS_ZONE]
   --driver, -d "none"
        Driver to create machine with. [$MACHINE_DRIVER]
   --engine-env [--engine-env option --engine-env option]
        Specify environment variables to set in the engine
   --engine-insecure-registry [--engine-insecure-registry option --engine-insecure-registry option
     Specify insecure registries to allow with the created engine
   --engine-install-url "https://get.docker.com"
        Custom URL to use for engine installation [$MACHINE_DOCKER_INSTALL_URL]
   --engine-label [--engine-label option --engine-label option]
        Specify labels for the created engine
   --engine-opt [--engine-opt option --engine-opt option]
        Specify arbitrary flags to include with the created engine in the form flag=value
   --engine-registry-mirror [--engine-registry-mirror option --engine-registry-mirror option]
        Specify registry mirrors to use [$ENGINE_REGISTRY_MIRROR]
   --engine-storage-driver
        Specify a storage driver to use with the engine
   --swarm
        Configure Machine to join a Swarm cluster
   --swarm-addr
        addr to advertise for Swarm (default: detect and use the machine IP)
   --swarm-discovery
        Discovery service to use with Swarm
   --swarm-experimental
        Enable Swarm experimental features
   --swarm-host "tcp://0.0.0.0:3376"
        ip/socket to listen on for Swarm master
   --swarm-image "swarm:latest"
        Specify Docker image to use for Swarm [$MACHINE_SWARM_IMAGE]
   --swarm-join-opt [--swarm-join-opt option --swarm-join-opt option]
```

```
Options:

   --amazonec2-access-key
        AWS Access Key [$AWS_ACCESS_KEY_ID]
   --amazonec2-ami
        AWS machine image [$AWS_AMI]
   --amazonec2-block-duration-minutes "0"
        AWS spot instance duration in minutes (60, 120, 180, 240, 300, or 360)
   --amazonec2-device-name "/dev/sda1"
        AWS root device name [$AWS_DEVICE_NAME]
   --amazonec2-endpoint
        Optional endpoint URL (hostname only or fully qualified URI) [$AWS_ENDPOINT]
   --amazonec2-iam-instance-profile
        AWS IAM Instance Profile [$AWS_INSTANCE_PROFILE]
   --amazonec2-insecure-transport
        Disable SSL when sending requests [$AWS_INSECURE_TRANSPORT]
   --amazonec2-instance-type "t2.micro"
        AWS instance type [$AWS_INSTANCE_TYPE]
   --amazonec2-keypair-name
        AWS keypair to use; requires --amazonec2-ssh-keypath [$AWS_KEYPAIR_NAME]
   --amazonec2-monitoring
        Set this flag to enable CloudWatch monitoring
   --amazonec2-open-port [--amazonec2-open-port option --amazonec2-open-port option]
        Make the specified port number accessible from the Internet
   --amazonec2-private-address-only
        Only use a private IP address
   --amazonec2-region "us-east-1"
        AWS region [$AWS_DEFAULT_REGION]
   --amazonec2-request-spot-instance
        Set this flag to request spot instance
   --amazonec2-retries "5"
        Set retry count for recoverable failures (use -1 to disable)
   --amazonec2-root-size "16"
        AWS root disk size (in GB) [$AWS_ROOT_SIZE]
   --amazonec2-secret-key
        AWS Secret Key [$AWS_SECRET_ACCESS_KEY]
   --amazonec2-security-group [--amazonec2-security-group option --amazonec2-security-group option]
     AWS VPC security group [$AWS_SECURITY_GROUP]
   --amazonec2-session-token
        AWS Session Token [$AWS_SESSION_TOKEN]
   --amazonec2-spot-price "0.50"
        AWS spot instance bid price (in dollar)
   --amazonec2-ssh-keypath
        SSH Key for Instance [$AWS_SSH_KEYPATH]
   --amazonec2-ssh-user "ubuntu"
```

```
   --amazonec2-subnet-id
        AWS VPC subnet id [$AWS_SUBNET_ID]
   --amazonec2-tags
        AWS Tags (e.g. key1,value1,key2,value2) [$AWS_TAGS]
   --amazonec2-use-ebs-optimized-instance
        Create an EBS optimized instance
   --amazonec2-use-private-address
        Force the usage of private IP address
   --amazonec2-userdata
        path to file with cloud-init user data [$AWS_USERDATA]
   --amazonec2-volume-type "gp2"
        Amazon EBS volume type [$AWS_VOLUME_TYPE]
   --amazonec2-vpc-id
        AWS VPC id [$AWS_VPC_ID]
   --amazonec2-zone "a"
        AWS zone for instance (i.e. a,b,c,d,e) [$AWS_ZONE]
   --driver, -d "none"
        Driver to create machine with. [$MACHINE_DRIVER]
   --engine-env [--engine-env option --engine-env option]
        Specify environment variables to set in the engine
   --engine-insecure-registry [--engine-insecure-registry option --engine-insecure-registry optior
     Specify insecure registries to allow with the created engine
   --engine-install-url "https://get.docker.com"
        Custom URL to use for engine installation [$MACHINE_DOCKER_INSTALL_URL]
   --engine-label [--engine-label option --engine-label option]
        Specify labels for the created engine
   --engine-opt [--engine-opt option --engine-opt option]
        Specify arbitrary flags to include with the created engine in the form flag=value
   --engine-registry-mirror [--engine-registry-mirror option --engine-registry-mirror option]
        Specify registry mirrors to use [$ENGINE_REGISTRY_MIRROR]
   --engine-storage-driver
        Specify a storage driver to use with the engine
   --swarm
        Configure Machine to join a Swarm cluster
   --swarm-addr
        addr to advertise for Swarm (default: detect and use the machine IP)
   --swarm-discovery
        Discovery service to use with Swarm
   --swarm-experimental
        Enable Swarm experimental features
   --swarm-host "tcp://0.0.0.0:3376"
        ip/socket to listen on for Swarm master
   --swarm-image "swarm:latest"
        Specify Docker image to use for Swarm [$MACHINE_SWARM_IMAGE]
   --swarm-join-opt [--swarm-join-opt option --swarm-join-opt option]
```

(command-line arguments are a lousy DSL for describing infrastructure)

Terraform-as-a-Service

So let's come back to Terraform.

Terraform-as-a-Service

Terraform has a pretty good DSL for describing infrastructure (HCL, providers, resources) and it's almost purely declarative.

Terraform-as-a-Service

Thanks to HCL's support for variables and interpolation, Terraform configurations can be *parameterised*.

Terraform-as-a-Service

And once a configuration is parameterised, you can use it as a *template*.

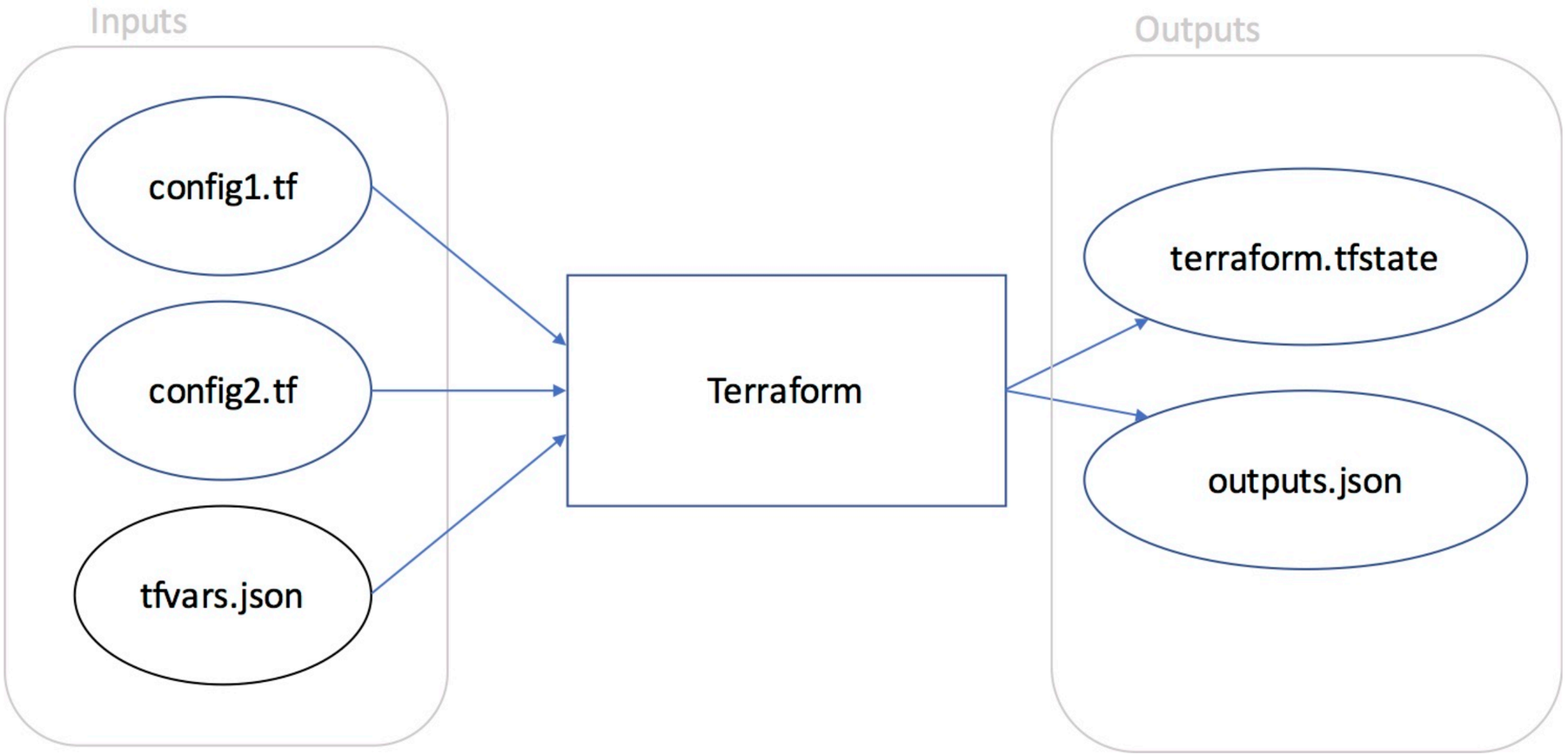
Terraform-as-a-Service

So Terraform is no longer simply a tool, but is now a reusable component.

So Terraform is no longer simply a tool, but is now a reusable component

It takes a configuration, together with one or more inputs (e.g. variables), applies the configuration, and then produces one or more outputs.

So Terraform is no longer simply a tool, but is now a reusable component



So Terraform is no longer simply a tool, but is now a reusable component

So if Terraform can be used as a template engine for infrastructure, then we have what we need to produce a more flexible driver for Docker Machine.

The Docker Machine driver copies the Terraform configuration, adds a file called `tfvars.json` containing variables to customise the configuration, and invokes Terraform.

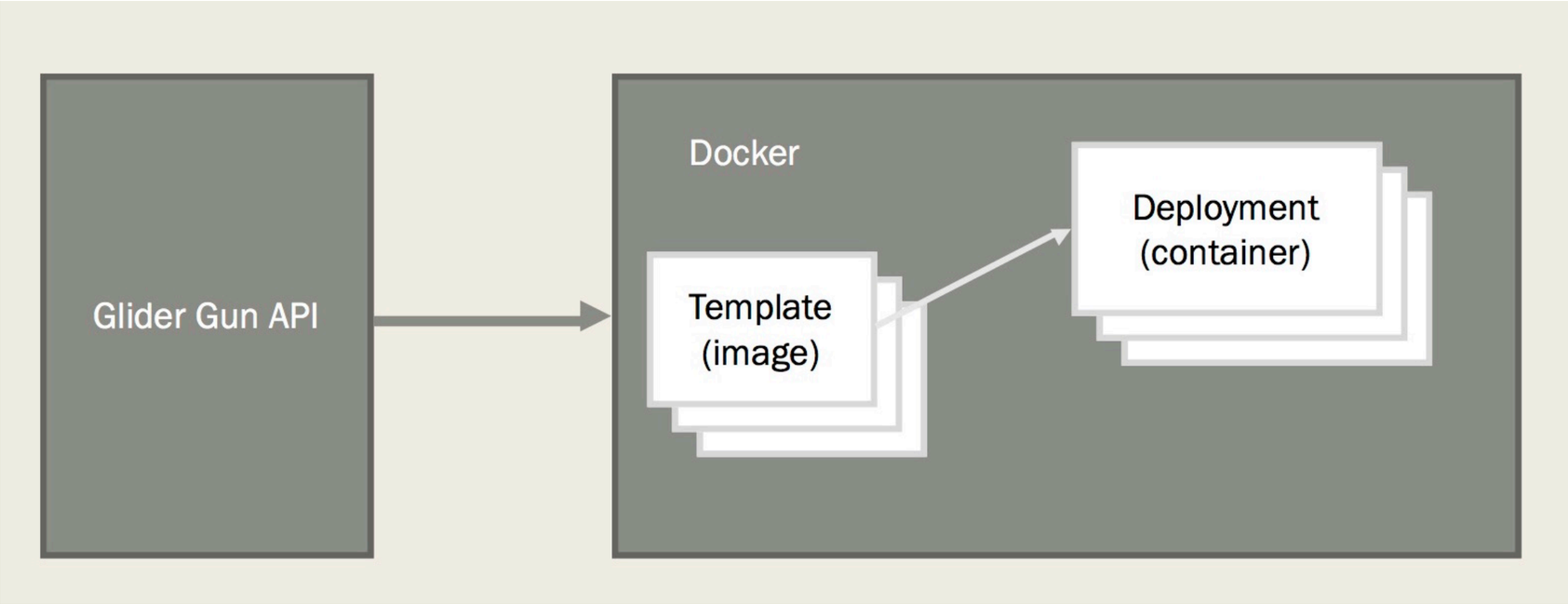
Finally, it reads the outputs from Terraform and passes information such as IP address and username (for SSH) back to Docker Machine (which uses it to install Docker).

Here's one

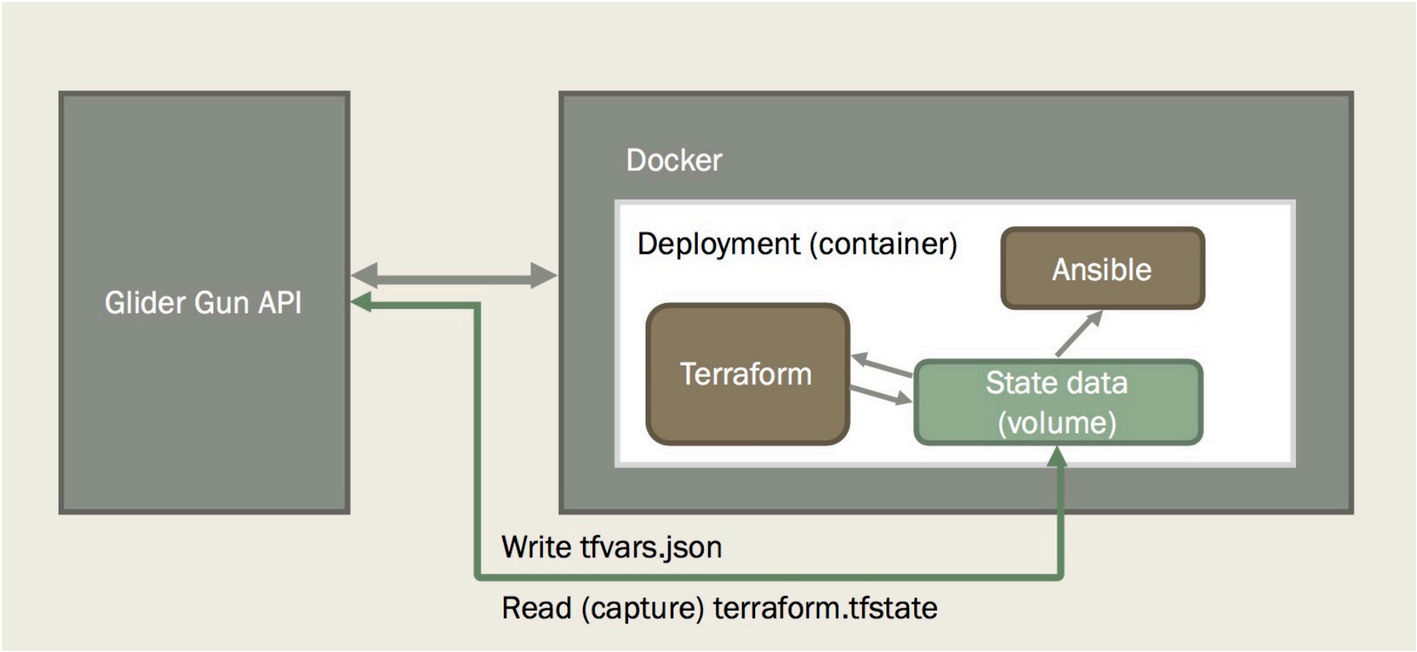
I prepared earlier

Terraform can be used like this for a variety of purposes; at a hackathon last year, we built an API-driven service catalog whose items deploy Terraform + Ansible configurations captured as Docker images.

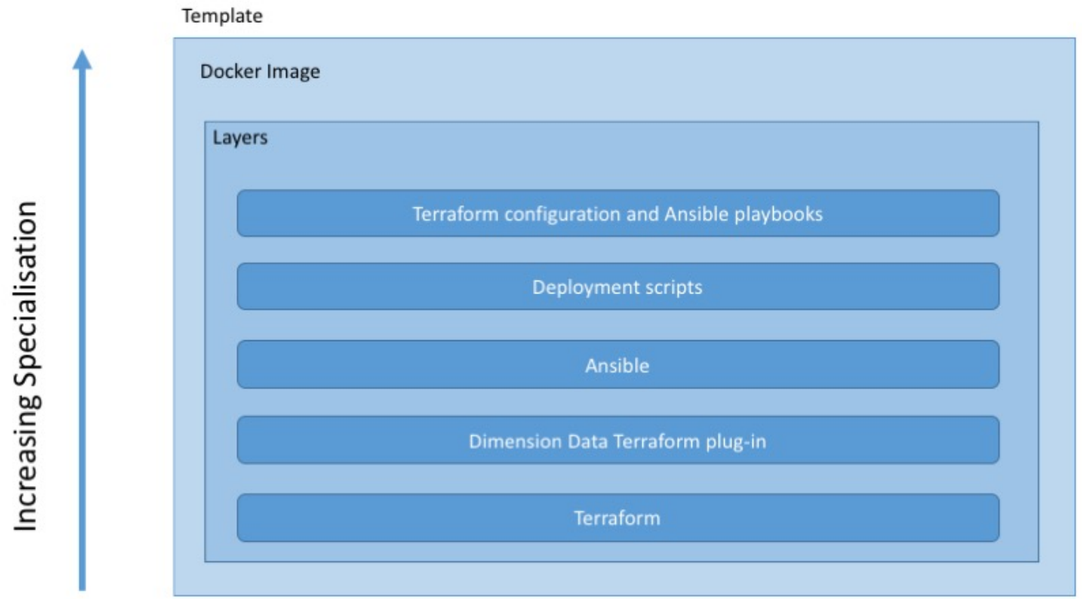
Terraform can be used like this for a variety of purposes; at a hackathon last year, we built an API-driven service catalog whose items deploy Terraform + Ansible configurations captured as Docker images.



Terraform can be used like this for a variety of purposes; at a hackathon last year, we built an API-driven service catalog whose items deploy Terraform + Ansible configurations captured as Docker images.



Terraform can be used like this for a variety of purposes; at a hackathon last year, we built an API-driven service catalog whose items deploy Terraform + Ansible configurations captured as Docker images.



Questions?

Links

github.com/tintoy/docker-machine-driver-terraform

github.com/DimensionDataResearch/glider-gun

Me

Adam Friedman (tintoy@tintoy.io)

https://github.com/tintoy

https://blog.tintoy.io/