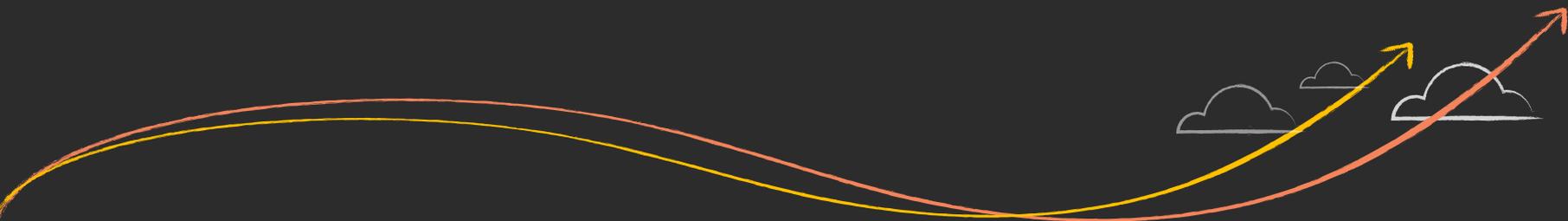




Kubernetes The Database

Jonathan Owens and Maryum Styles



Safe Harbor

This presentation and the information herein (including any information that may be incorporated by reference) is provided for informational purposes only and should not be construed as an offer, commitment, promise or obligation on behalf of New Relic, Inc. (“New Relic”) to sell securities or deliver any product, material, code, functionality, or other feature. Any information provided hereby is proprietary to New Relic and may not be replicated or disclosed without New Relic’s express written permission.

Such information may contain forward-looking statements within the meaning of federal securities laws. Any statement that is not a historical fact or refers to expectations, projections, future plans, objectives, estimates, goals, or other characterizations of future events is a forward-looking statement. These forward-looking statements can often be identified as such because the context of the statement will include words such as “believes,” “anticipates,” “expects” or words of similar import.

Actual results may differ materially from those expressed in these forward-looking statements, which speak only as of the date hereof, and are subject to change at any time without notice. Existing and prospective investors, customers and other third parties transacting business with New Relic are cautioned not to place undue reliance on this forward-looking information. The achievement or success of the matters covered by such forward-looking statements are based on New Relic’s current assumptions, expectations, and beliefs and are subject to substantial risks, uncertainties, assumptions, and changes in circumstances that may cause the actual results, performance, or achievements to differ materially from those expressed or implied in any forward-looking statement. Further information on factors that could affect such forward-looking statements is included in the filings New Relic makes with the SEC from time to time. Copies of these documents may be obtained by visiting New Relic’s Investor Relations website at ir.newrelic.com or the SEC’s website at www.sec.gov.

New Relic assumes no obligation and does not intend to update these forward-looking statements, except as required by law. New Relic makes no warranties, expressed or implied, in this presentation or otherwise, with respect to the information provided.



A black toaster is positioned on the left side of the countertop.

A white microwave is located on the countertop to the right of the toaster.

A roll of white paper towels is mounted on a holder on the countertop.

A bottle of wine or beer is visible on the countertop near the sink.

A yellow can, likely a beverage, is on the countertop.

A purple can is on the countertop.

A red can is on the countertop.

A bottle of dish soap is on the countertop near the sink.

Two knives are lying on the countertop near the sink.

A clear glass bowl is placed in the right basin of the sink.

A white plate with a floral pattern, a knife, and a fork is on the right side of the sink.

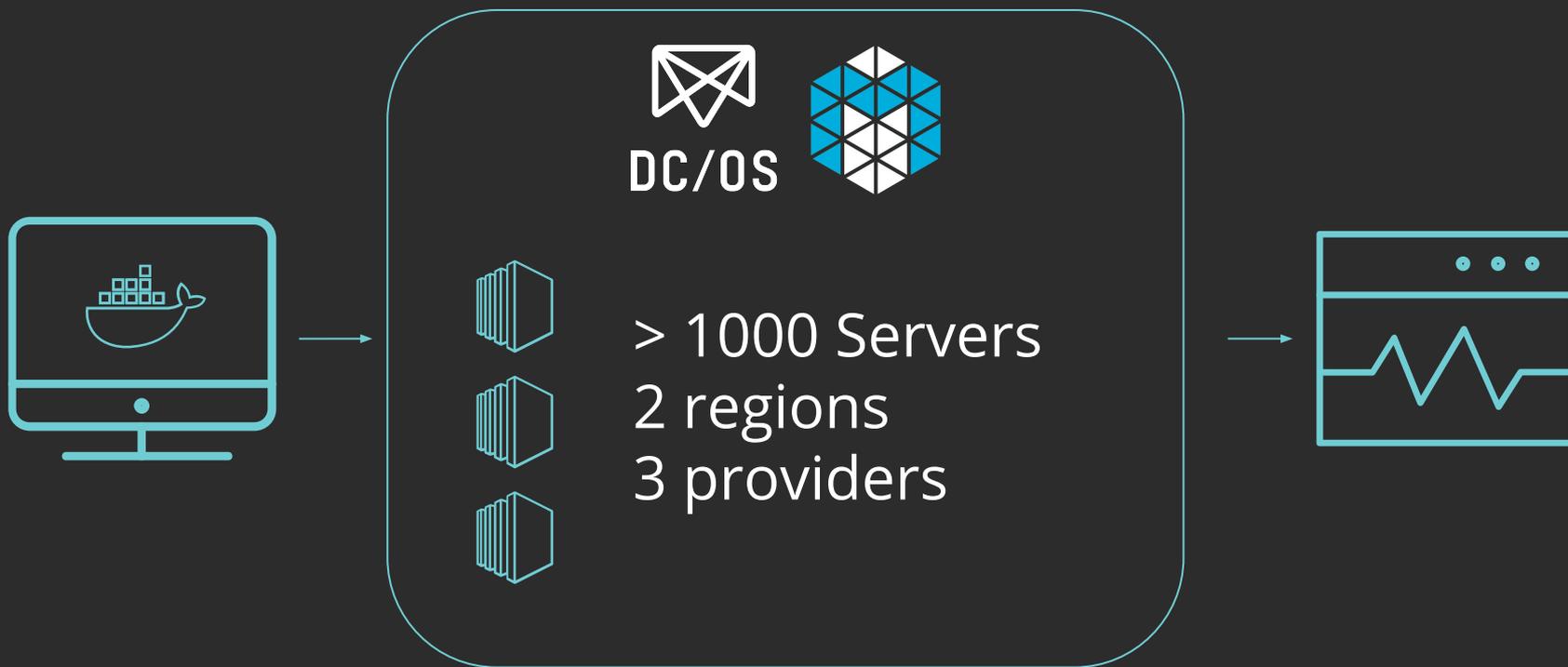
White cabinets with dark knobs are located on the left side of the kitchen.

The floor is made of dark wood planks.

A window is visible in the background, showing a view of the outdoors.



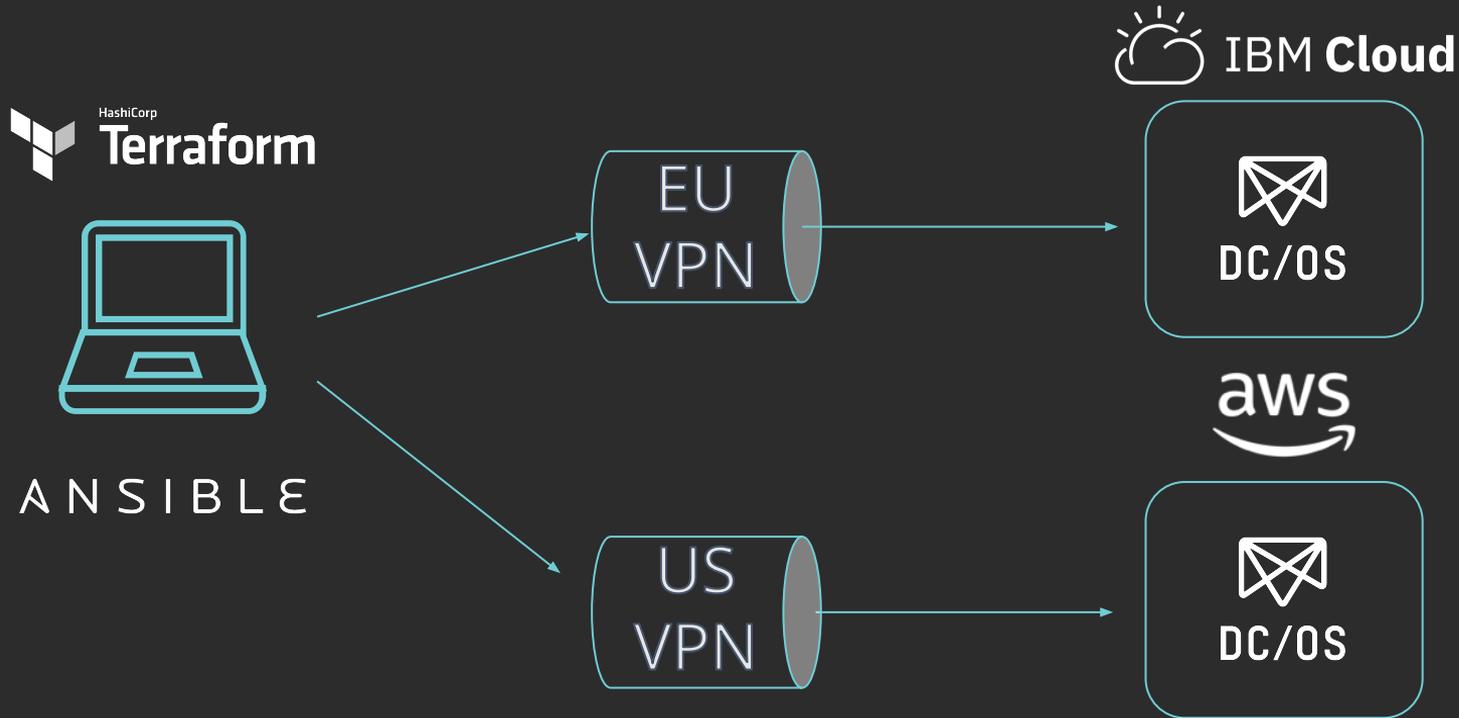
Containerized Application Fleet



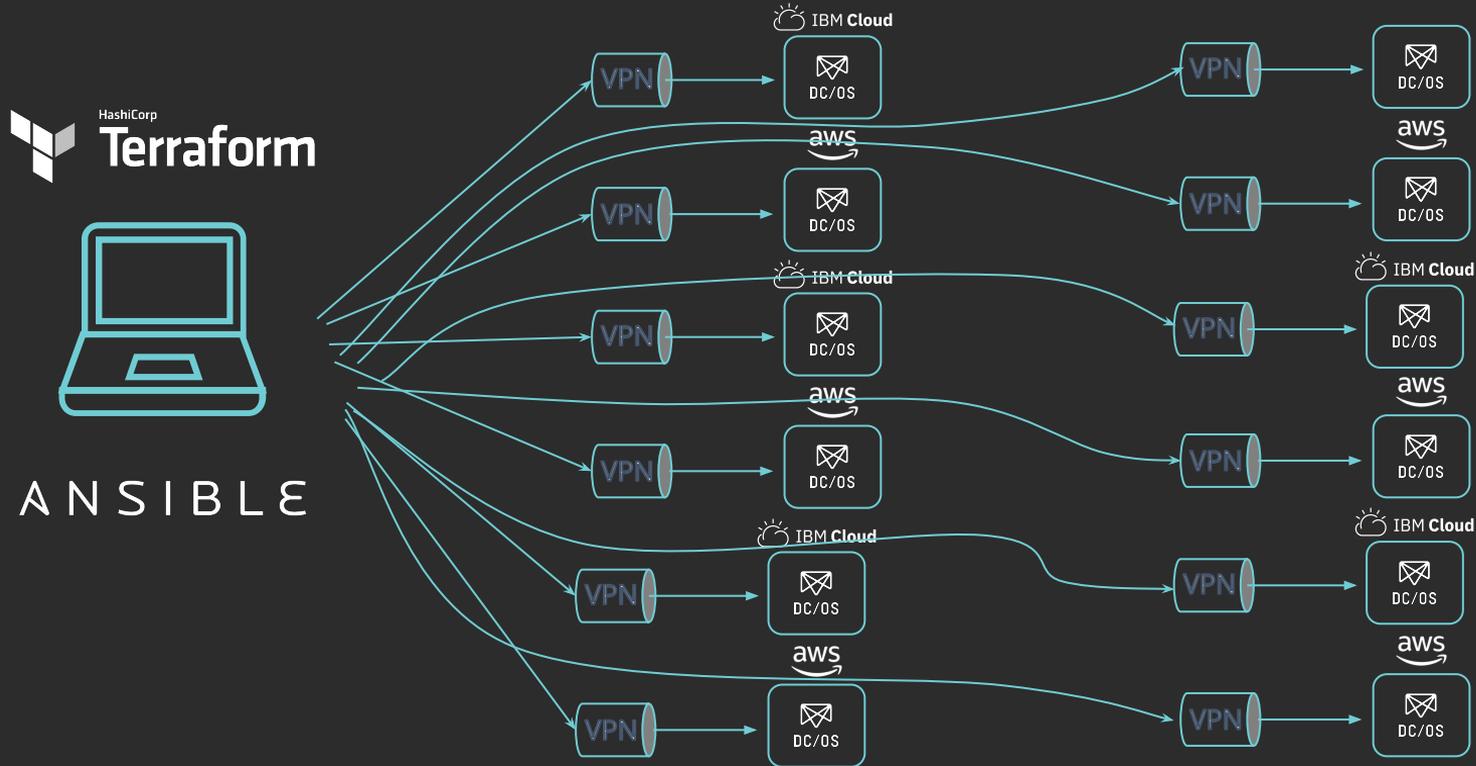
Container Fabric Management Architecture



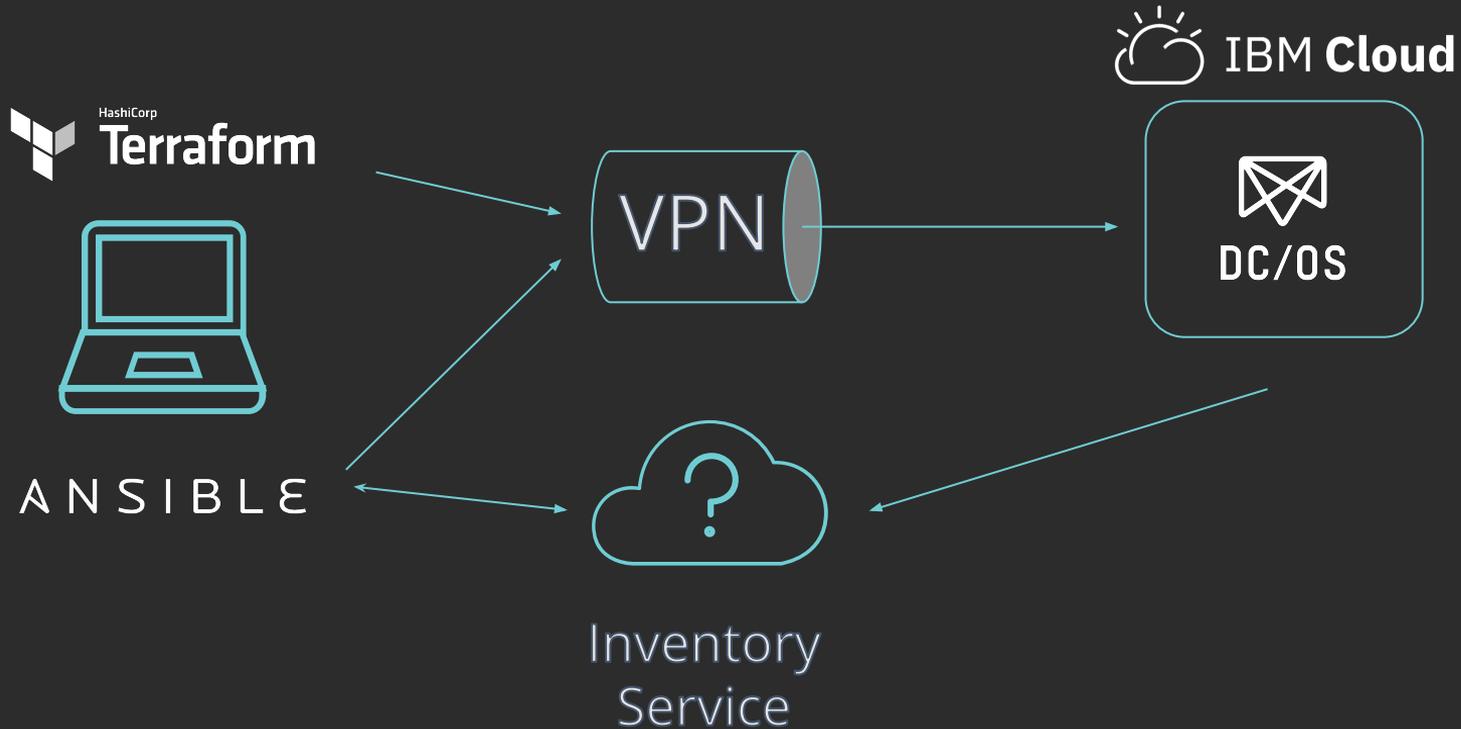
Container Fabric Regional Architecture



Architecture Forecast



Inventory, not Connectivity



Multi-Cloud Inventory



What could we use?

Terraform state?



Inventory
Service

- Not distributed
- No partial updates

Commercial DCIM?

- We don't have one
- Wrong abstraction

DC/OS?

What turned out to matter



Inventory
Service

- Single source of information
- Live worldwide updates
- Ansible inventory compatible CLI
- Production quality
- Locally meaningful representation

K8s



Kubernetes Features

Centralized configuration service

Flexible configuration updates from many clients. Set cluster configuration from supported client libraries in a declarative, secure way.

Storage orchestration

Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as [GCP](#) or [AWS](#), or a network storage system such as NFS, iSCSI, Gluster, Ceph, Cinder, or Flocker.

Automated rollouts and rollbacks

Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

Batch execution

In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.

Automatic binpacking

Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.

Self-healing

Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Secret and configuration management

Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

Horizontal scaling

Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

K1s



Kubernetes Features

Centralized configuration service

Flexible configuration updates from many clients. Set cluster configuration from supported client libraries in a declarative, secure way.

K1s

kube-apiserver

Synopsis

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

```
kube-apiserver [flags]
```

API Server Objects

Core Objects



Service



Deployment



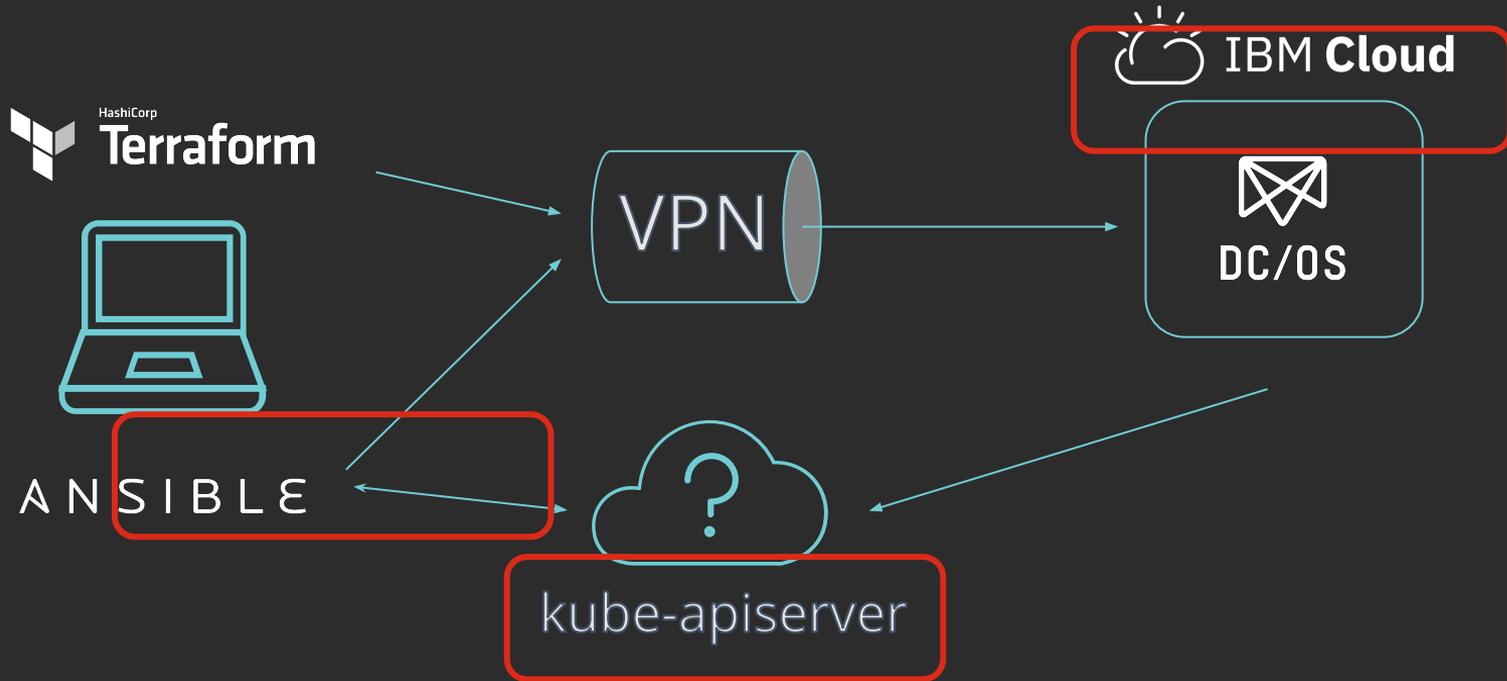
StatefulSet

Custom Objects



CustomResource

Custom Resources Everywhere

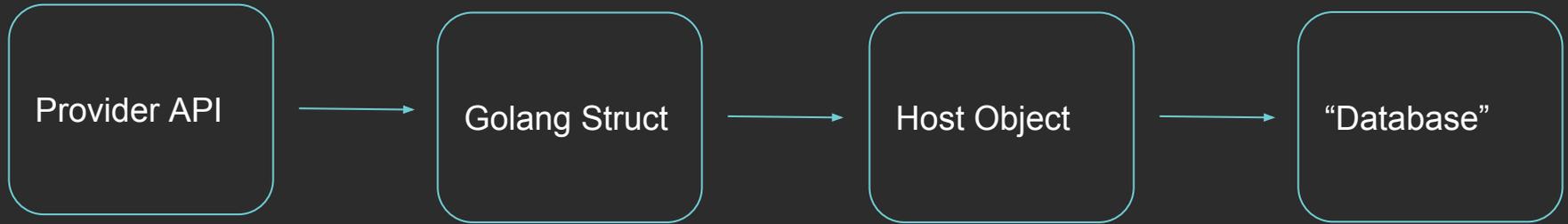




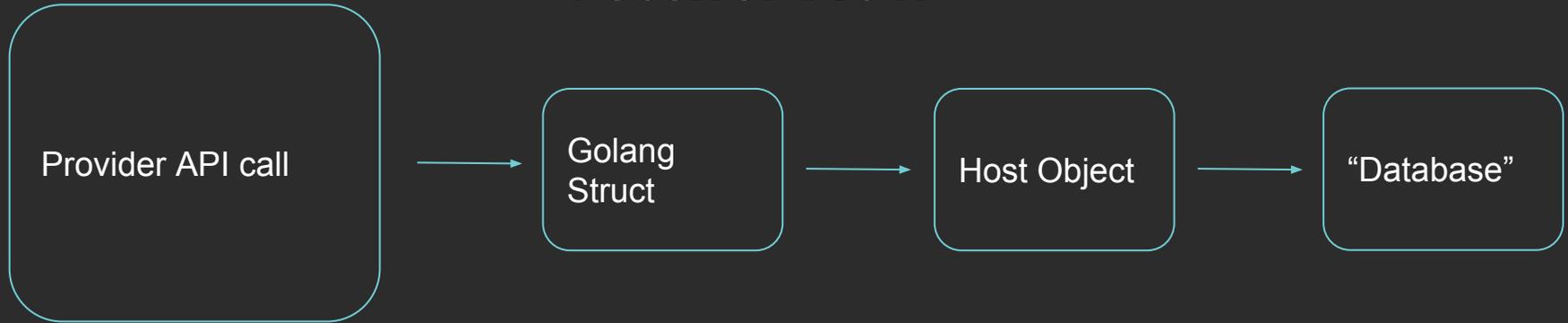
Implementation

Maryum Styles

Fetcher (Controller) Overview



Fetcher Flow

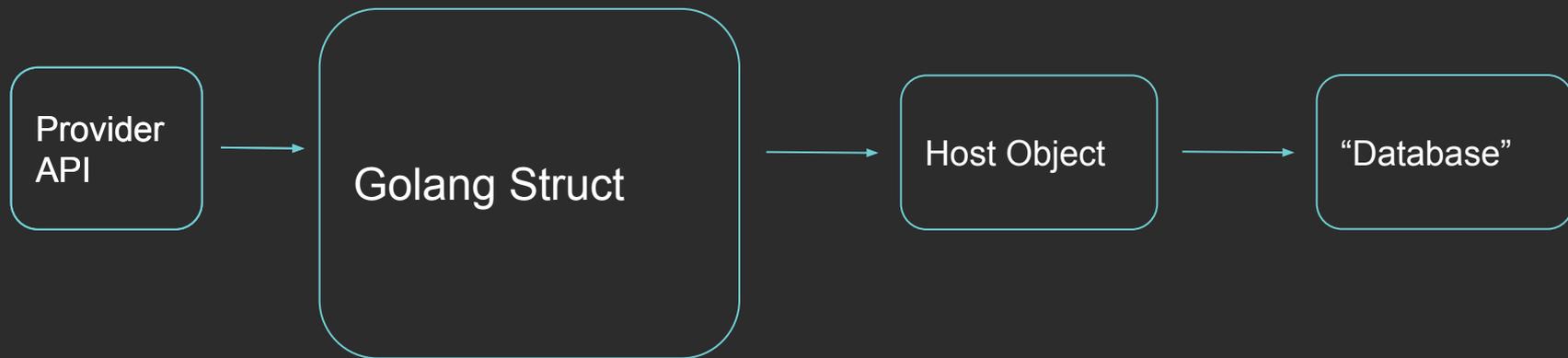


IBM Cloud



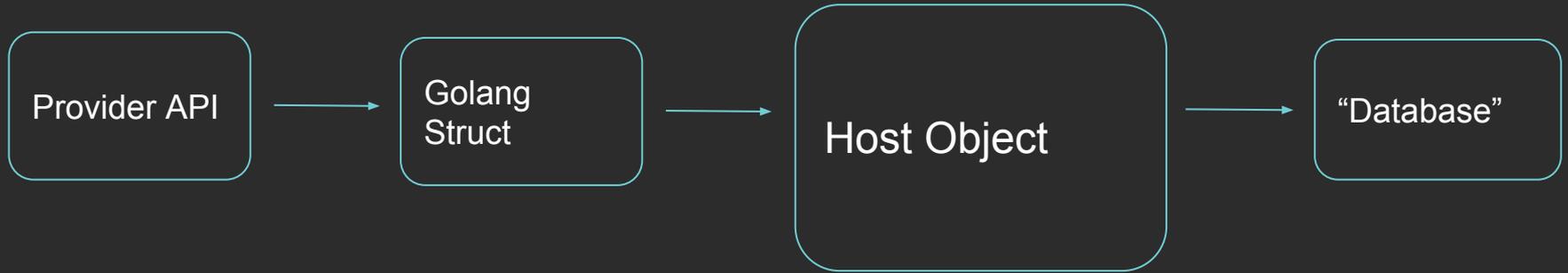
Datacenter

Fetcher Flow



Json responses from Provider APIs are stored as structs specific to each provider

Fetcher Flow



Host Objects are Custom Resources in Kubernetes

Creating a Custom Resource

- create Custom Resource Definitions (CRDs)
- create new object's struct fields
- update files with new struct name
- run code generation
- apply CRDs

reference: <https://kubernetes.io/docs/tasks/access-kubernetes-api/custom-resources/custom-resource-definitions/>

Host CRD

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: hosts.alpha.nr-ops.net
spec:
  group: alpha.nr-ops.net
  version: v1
  scope: Cluster
  names:
    plural: hosts
    singular: host
    kind: Host
    shortNames:
    - ho
```

Host Struct

```
type Host struct {  
    metav1.TypeMeta    `json:",inline"`  
    metav1.ObjectMeta  `json:"metadata,omitempty"`  
  
    Spec    HostSpec    `json:"spec"`  
    Status  HostStatus  `json:"status"`  
}
```

Host Object

Host Labels

hardware type
region
cluster name
environment

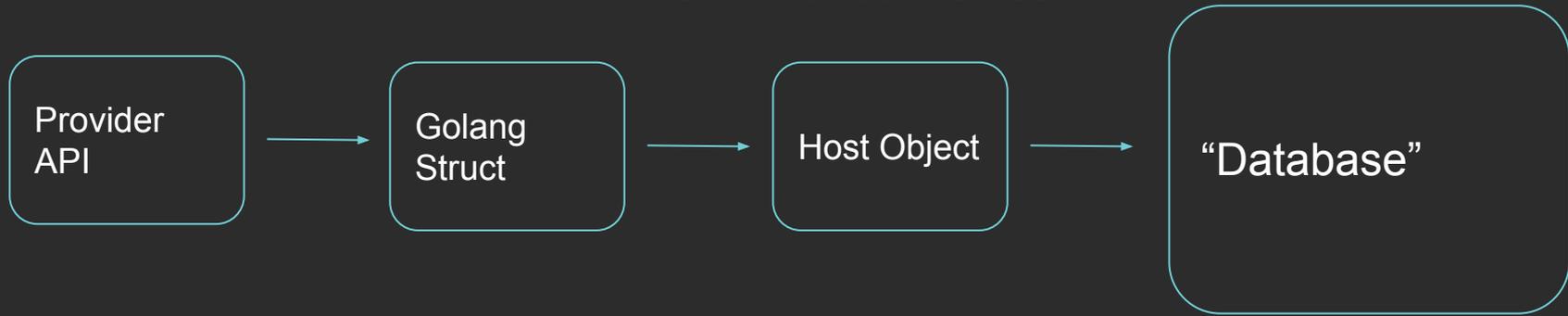
Used to find a host

Host Annotations

ansible variables
provider groups
ansible visibility

Used to store data about a host

Fetcher Flow



The kubernetes host objects is stored via an update or create call to the Kube API

The Deploy Story



Create CoreOS EC2 instance and ec2 ELB

ANSIBLE

Add Certs
Install Kube API
Install Controller-Manager



kubernetes

DC/OS



DC/OS

Using the “database”

run command:
kubectl get ho fra-sl-943337 -o yaml

KUBE API FORMAT

```
apiVersion: alpha.nr-ops.net/v1
kind: Host
  labels:
    availabilityZone: bcr01.fra02
    clusterName: fra1a
    env: eu-production
    os: coreos
    provider: softlayer
    region: eu
    role: log
    sla: high_priority
    ...
name: fra-sl-943337
```

Using the “database”

ANSIBLE FORMAT

```
"nr_role=log": {
  "hosts":
  ["cf-log-943337-fra1a.r112.eu.nr-ops.net"],
  "vars": {"nr_role": "log"}
},
"nr_cluster_name=fra1a": {
  "hosts":
  ["cf-log-943337-fra1a.r112.eu.nr-ops.net"],
  "vars": {"nr_cluster_name": "fra1a"}
},
```

Finalizers

Host CRD

```
apiVersion:  
apiextensions.k8s.io/v1beta1  
kind: CustomResourceDefinition  
metadata:  
  finalizers:  
  - finalizer.stable.example.com
```

Host Instance

```
apiVersion: alpha.nr-ops.net/v1  
kind: Host  
metadata:  
  finalizers:  
  - nr-ops.net/hostlifecycle
```



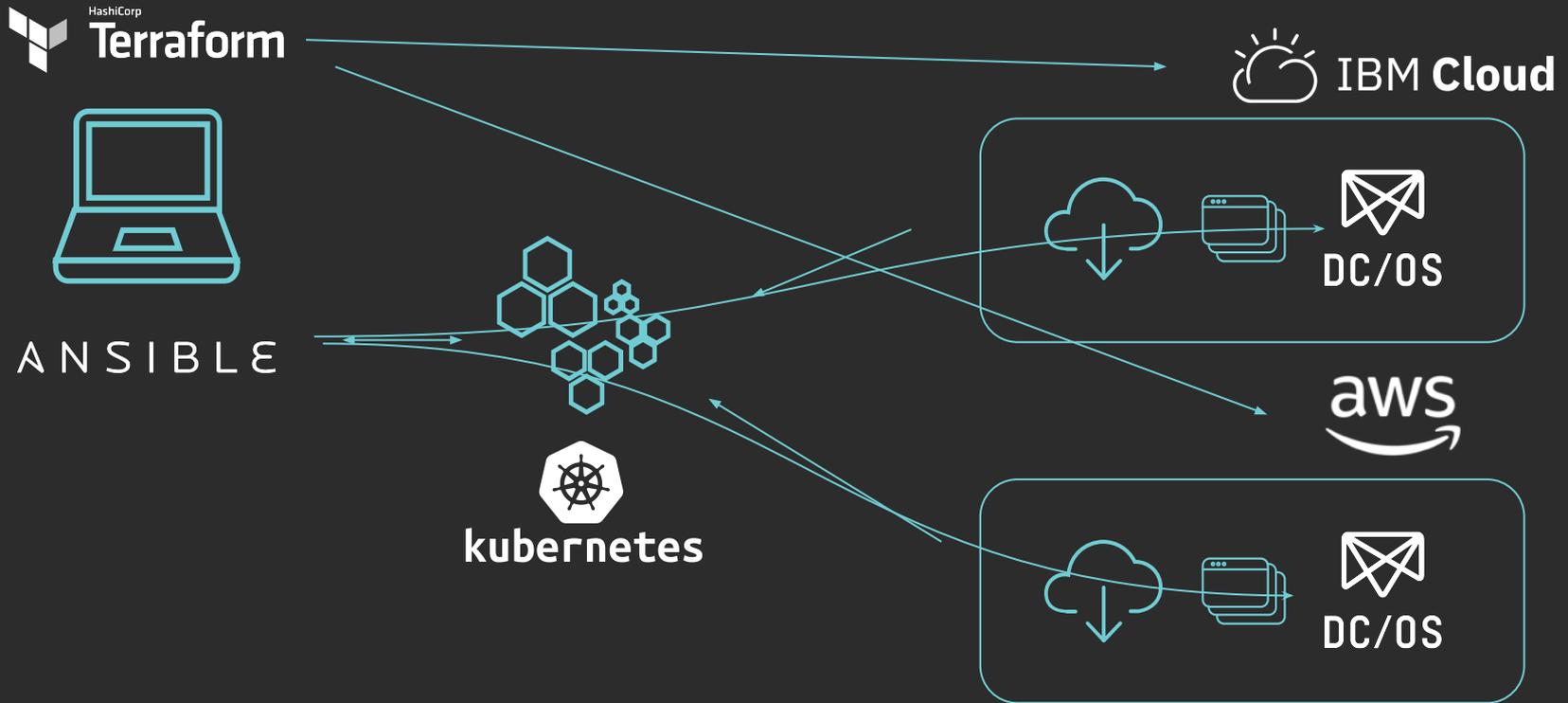
Patterns

Jonathan Owens

Old Management Architecture



Kubernetes API Inventory





A decorative graphic consisting of two wavy lines, one yellow and one orange, that curve upwards from left to right. An orange arrow points upwards and to the right, starting from the yellow line and ending near the top right. In the background, there are faint white outlines of clouds.

Thank You

Jonathan Owens | [@intjonathan](#) | jonathan@newrelic.com

Maryum Styles | mstyles@newrelic.com