New Relic®

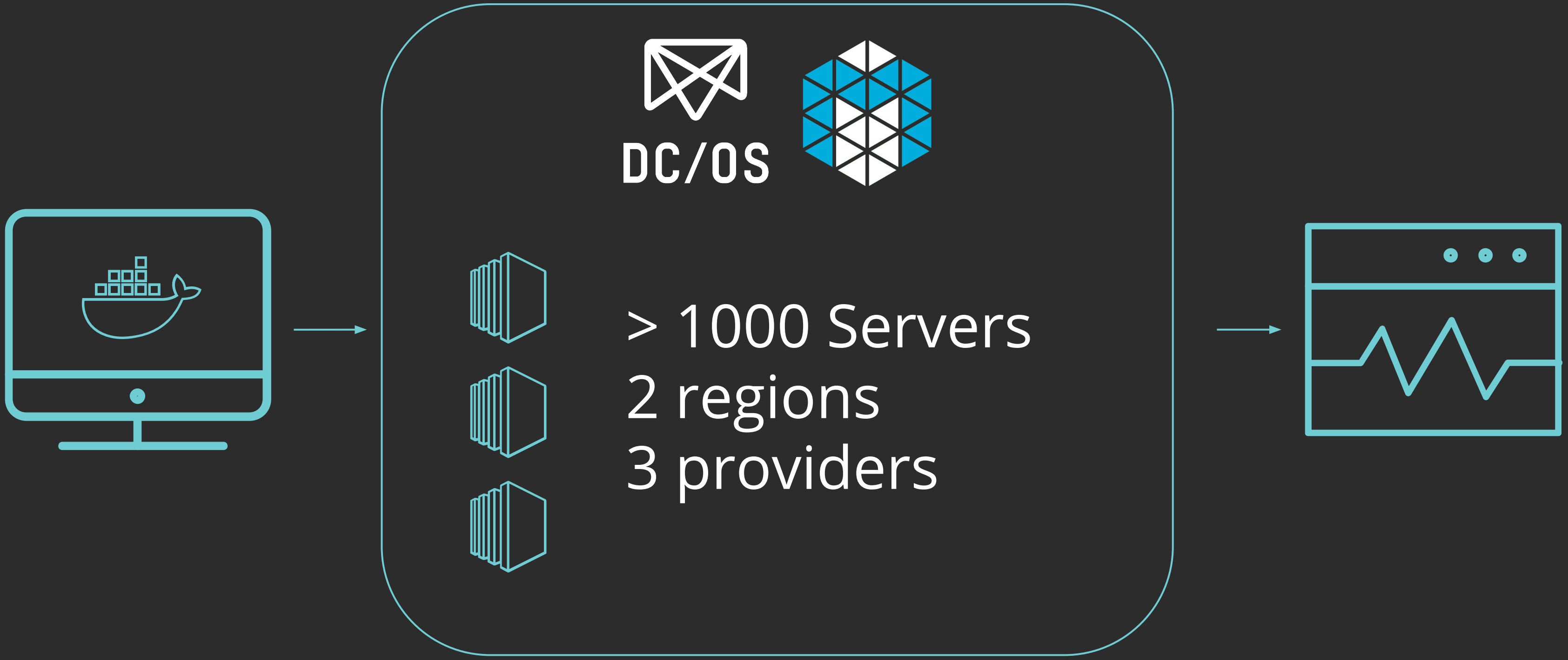
Kubernetes The Database

Jonathan Owens and Maryum Styles

Safe Harbor

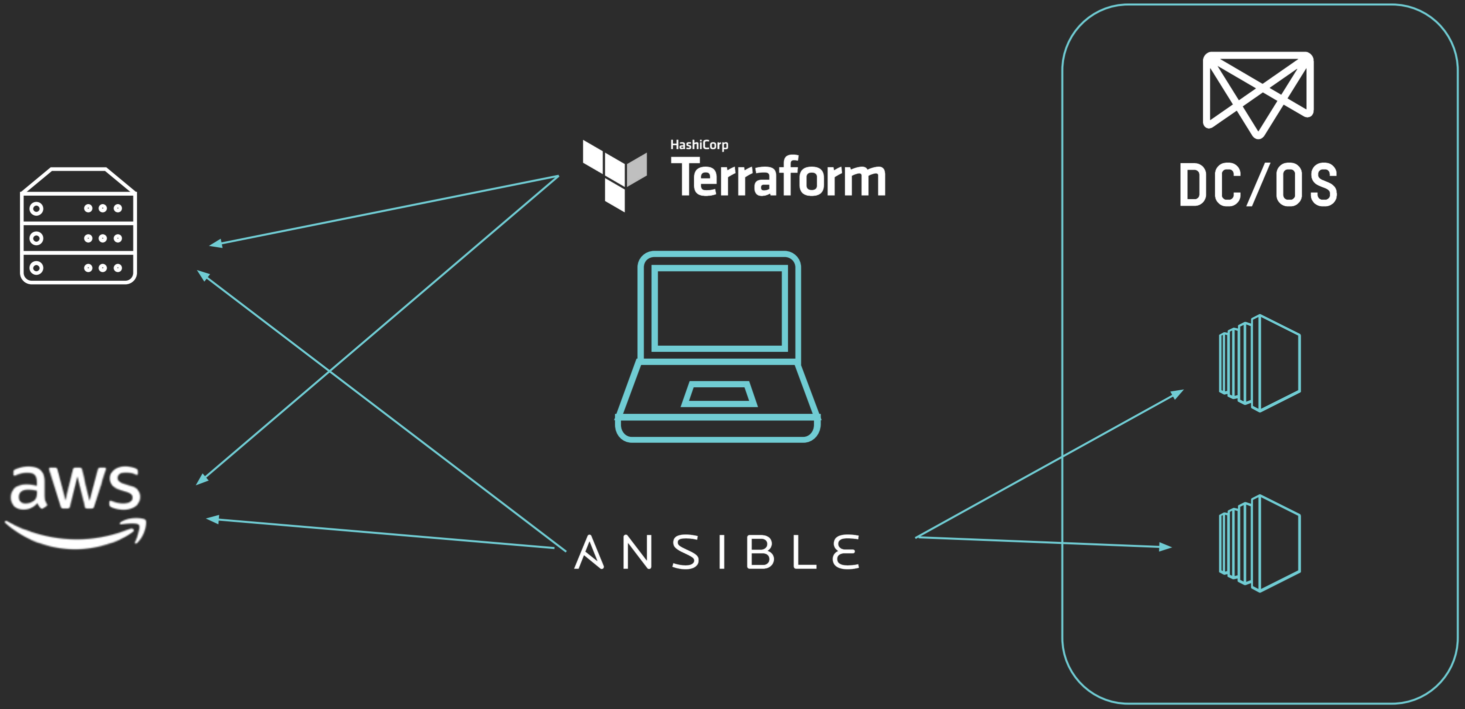
Recycling

Containerized Application Fleet

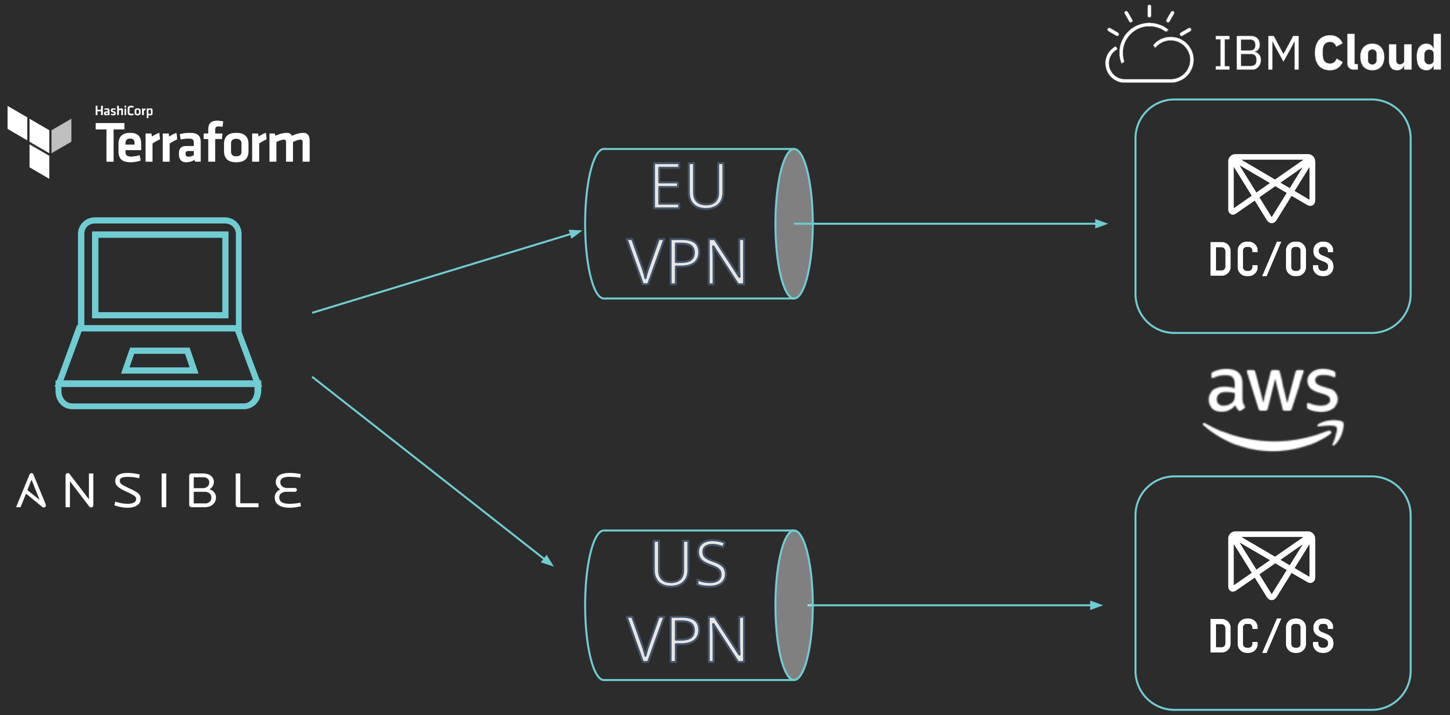


> 1000 Servers
2 regions
3 providers

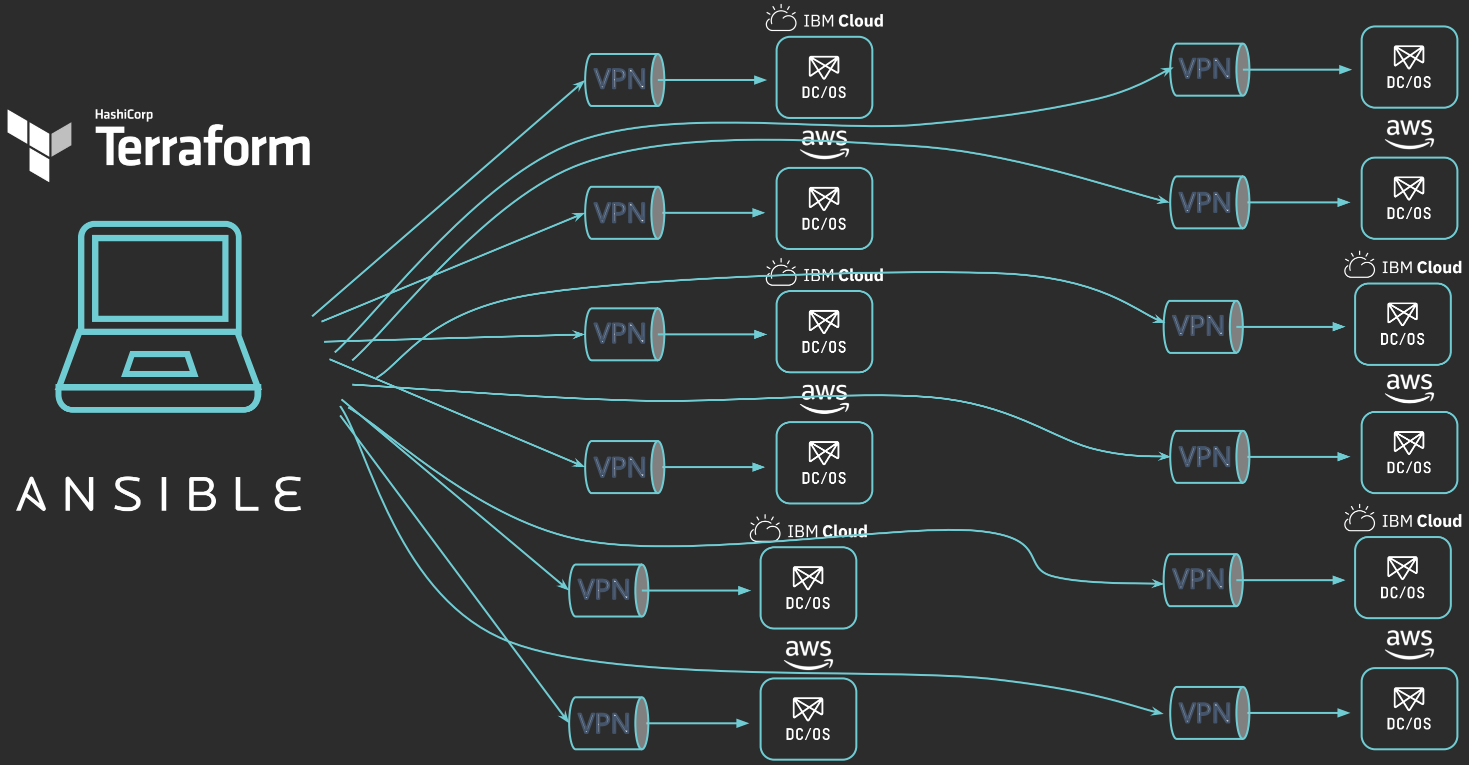
Container Fabric Management Architecture



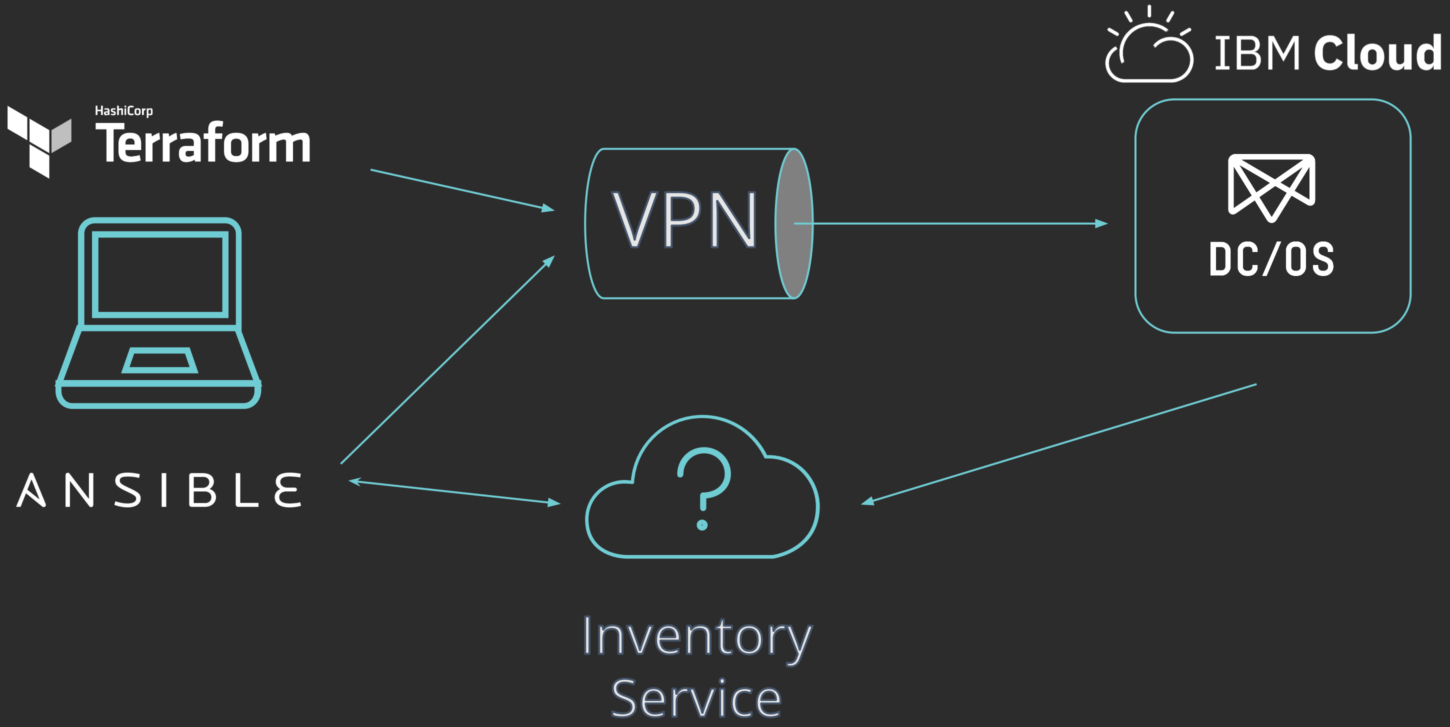
Container Fabric Regional Architecture



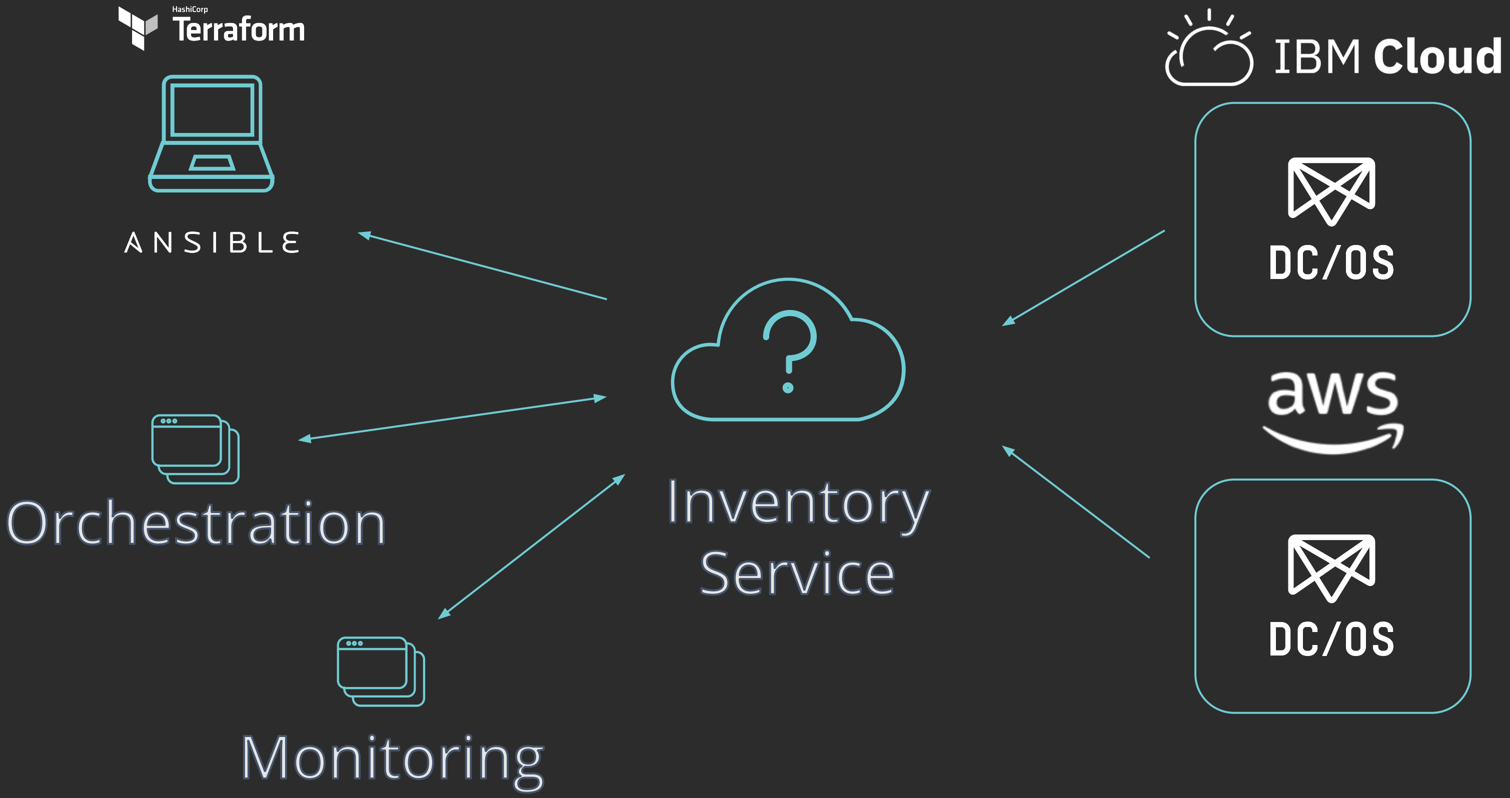
Architecture Forecast



Inventory, not Connectivity



Multi-Cloud Inventory



What could we use?

Inventory Service

Terraform state?

- Not distributed
- No partial updates

Commercial DCIM?

- We don't have one
- Wrong abstraction

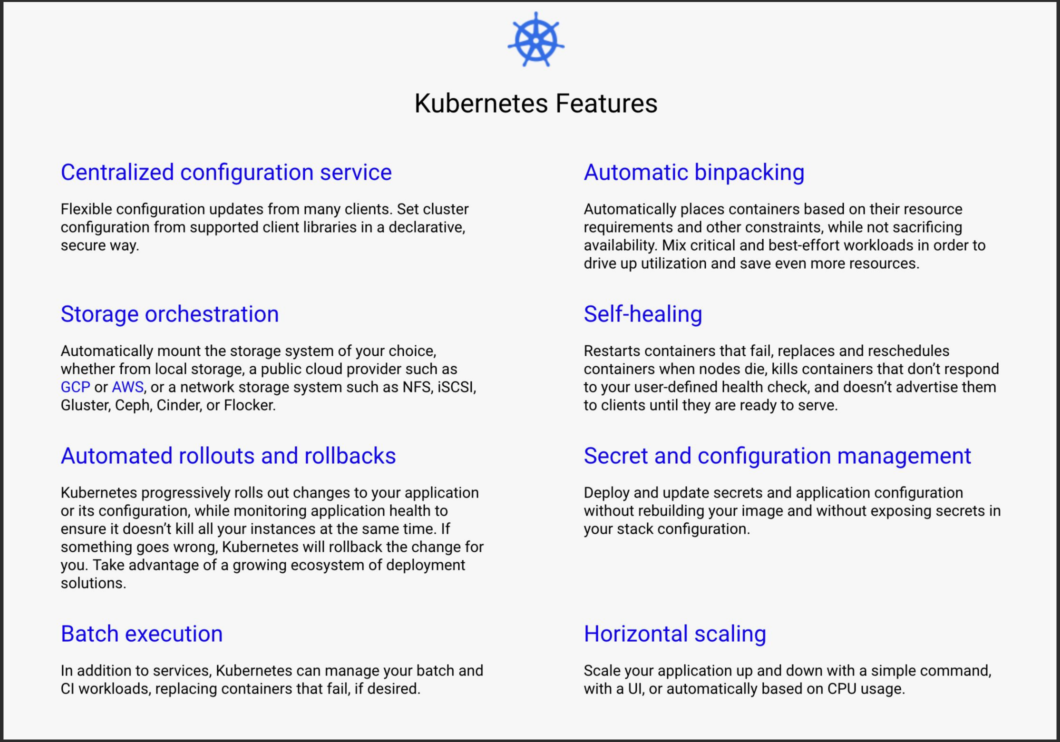
DC/OS?

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.

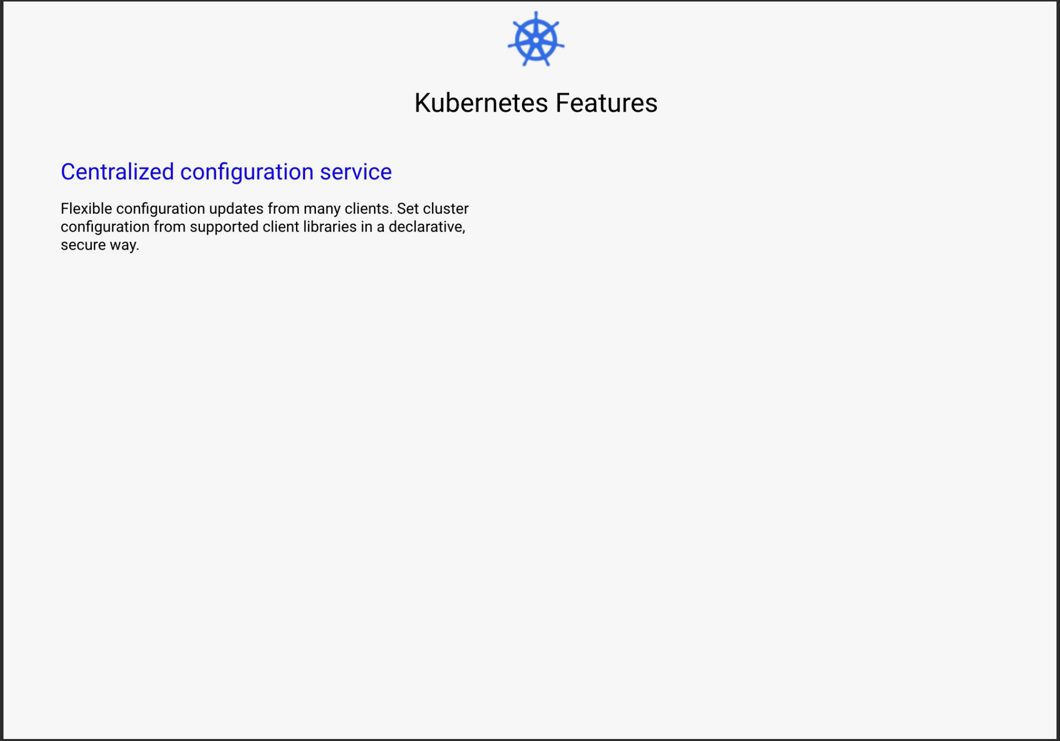
Secret and configuration management

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

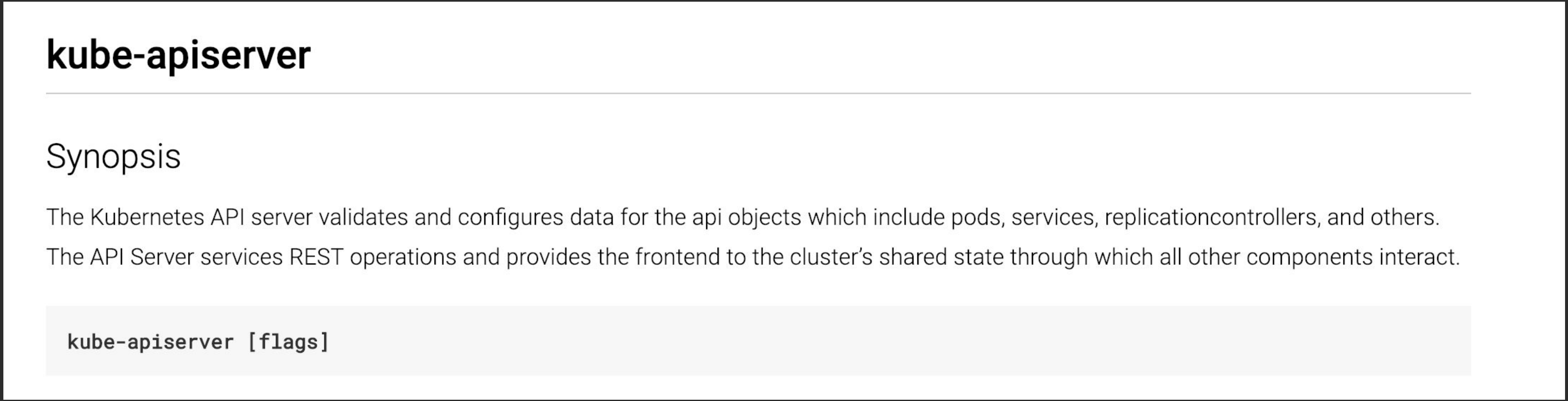


Kubernetes Features

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

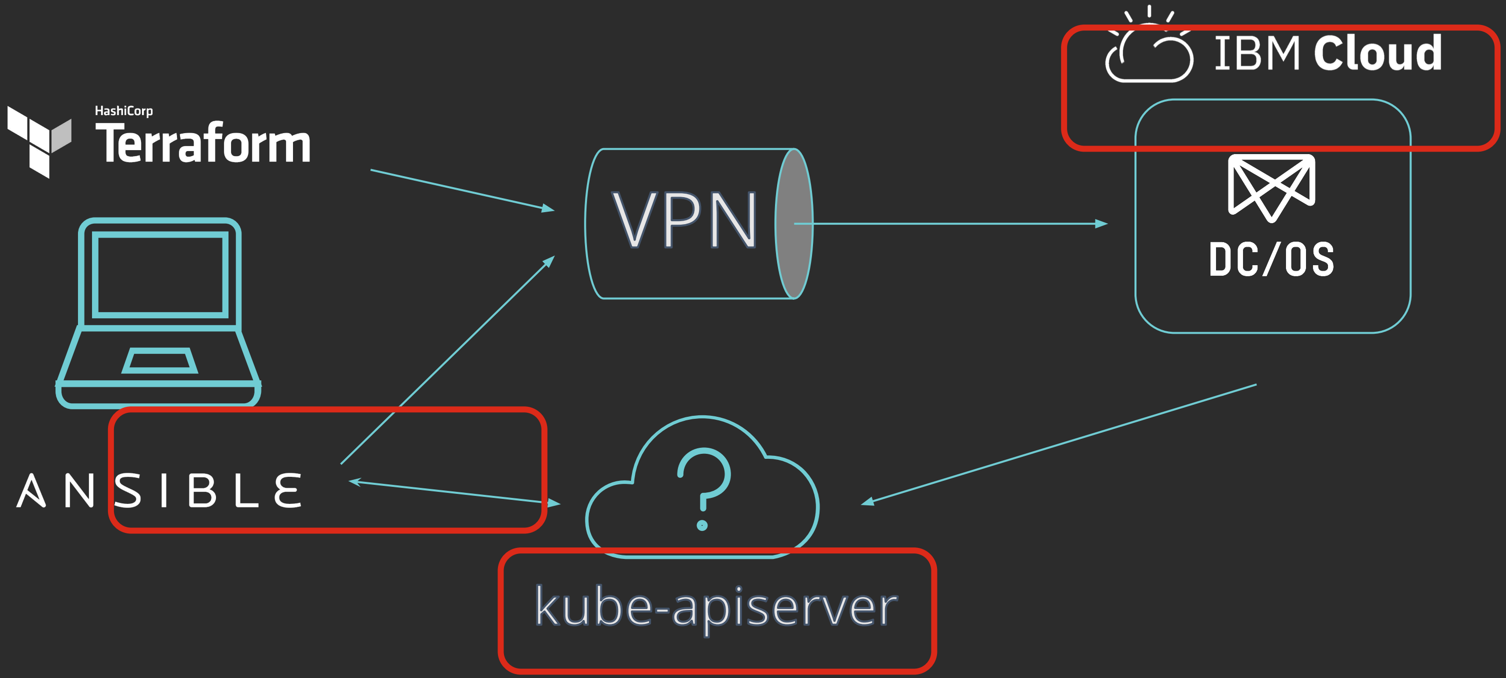
Deployment

StatefulSet

Custom Objects

CustomResource

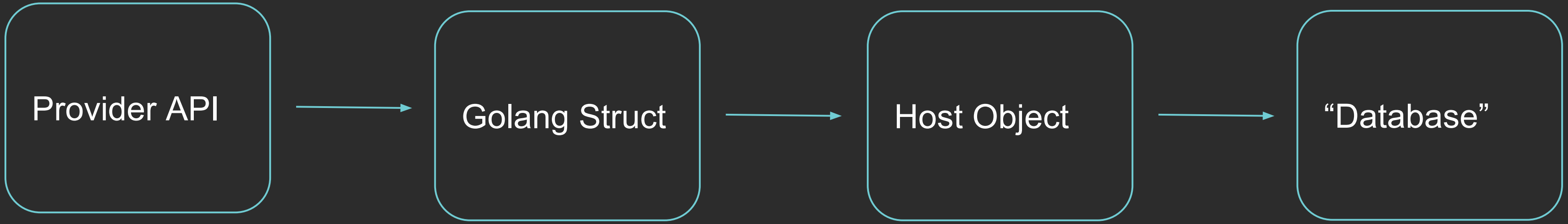
Custom Resources Everywhere



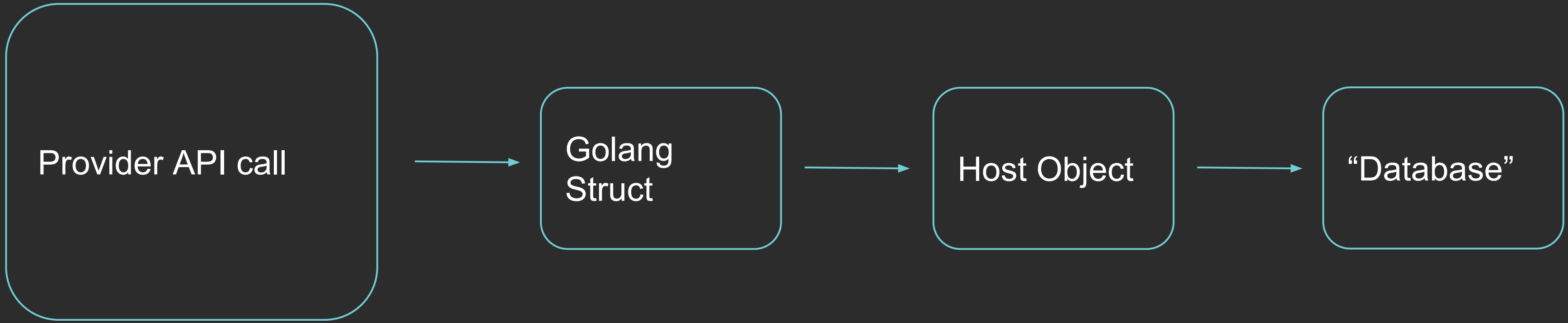
Implementation

Maryum Styles

Fetcher (Controller) Overview



Fetcher Flow

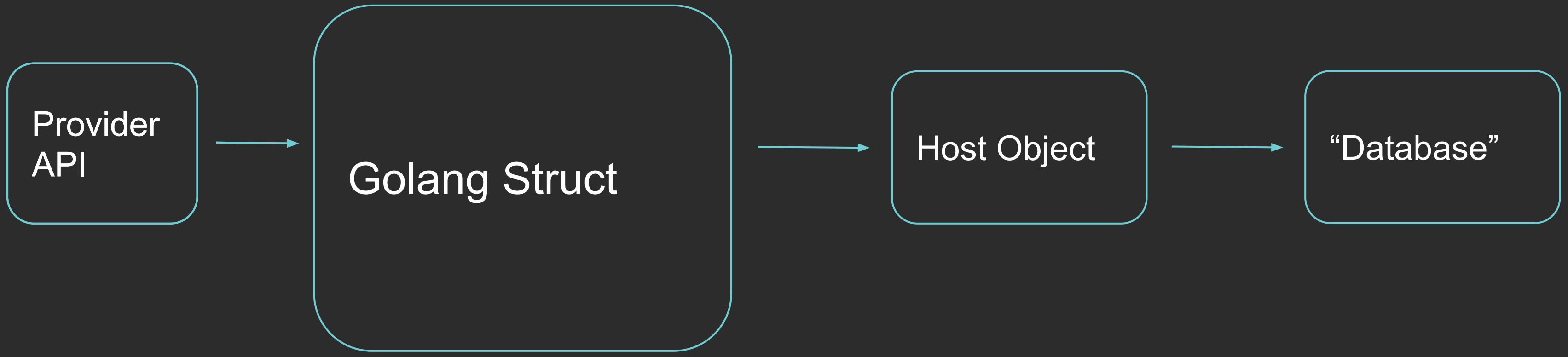


aws

IBM Cloud

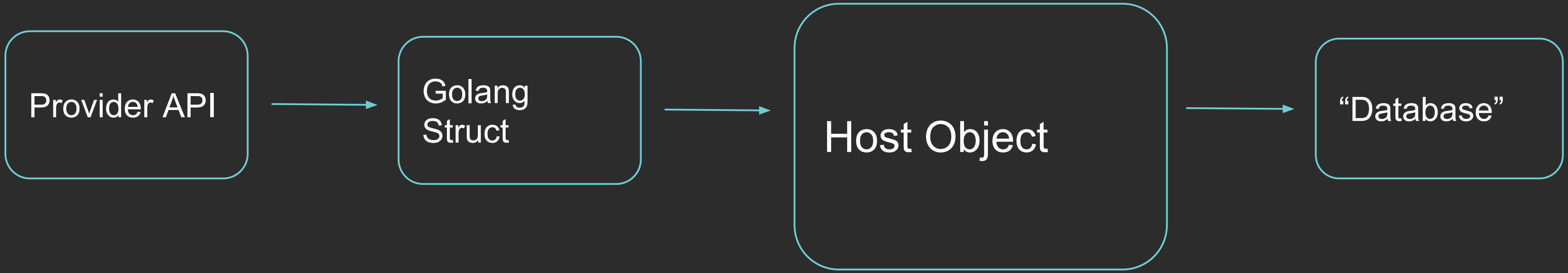
Datacenter

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

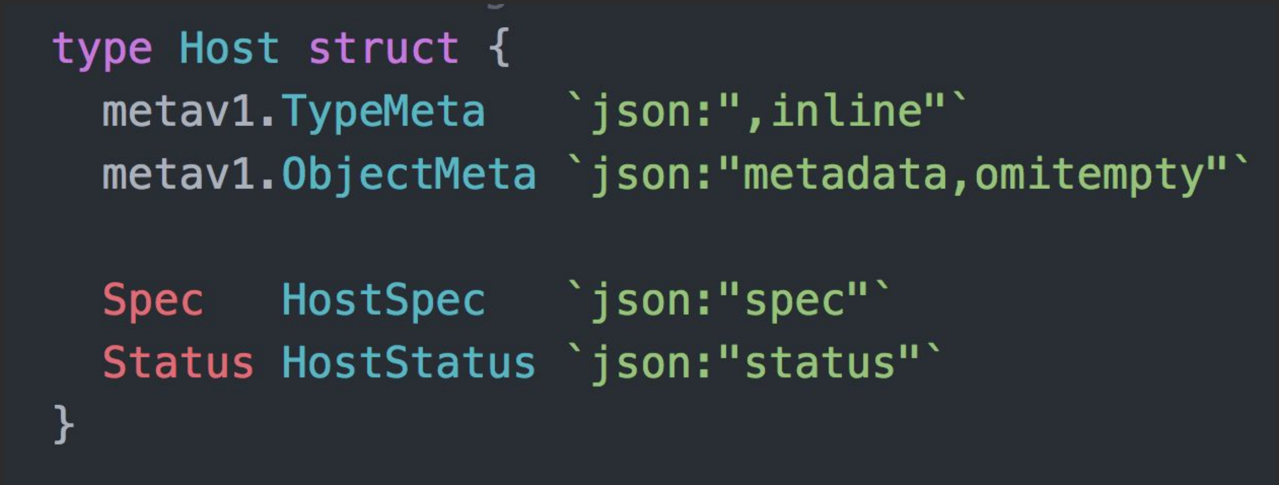
- create Cusotm Resource Definitions (CRDs)
- create new object’s struct fields
- update files with new struct name
- run code generation
- apply CRDs

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

Host CRD

```yaml
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



```go
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

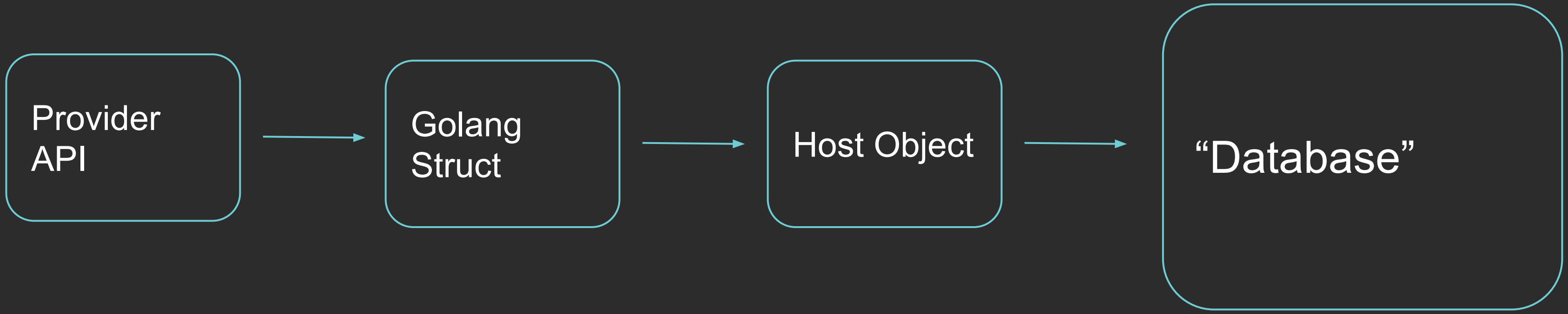
ansible variables

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

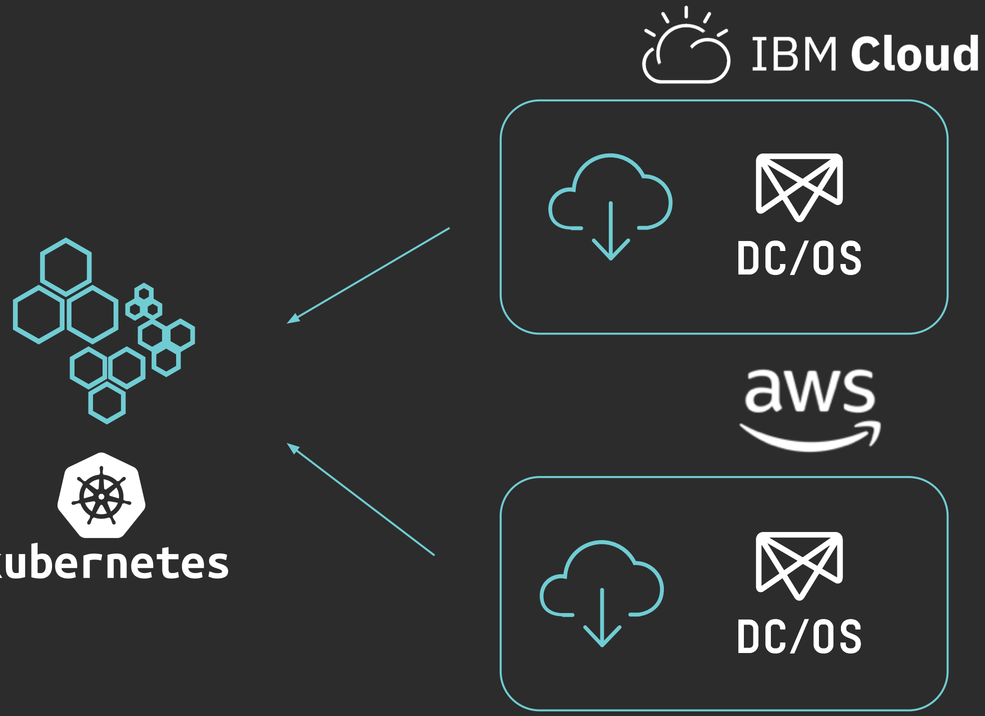
The Deploy Story

HashiCorp Terraform

Create CoreOS EC2
instance and ec2 ELB

ANSIBLE

Add Certs
Install Kube API
Install Controller-Manager



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"}
  },
```

New Relic.

Finalizers

Host CRD

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

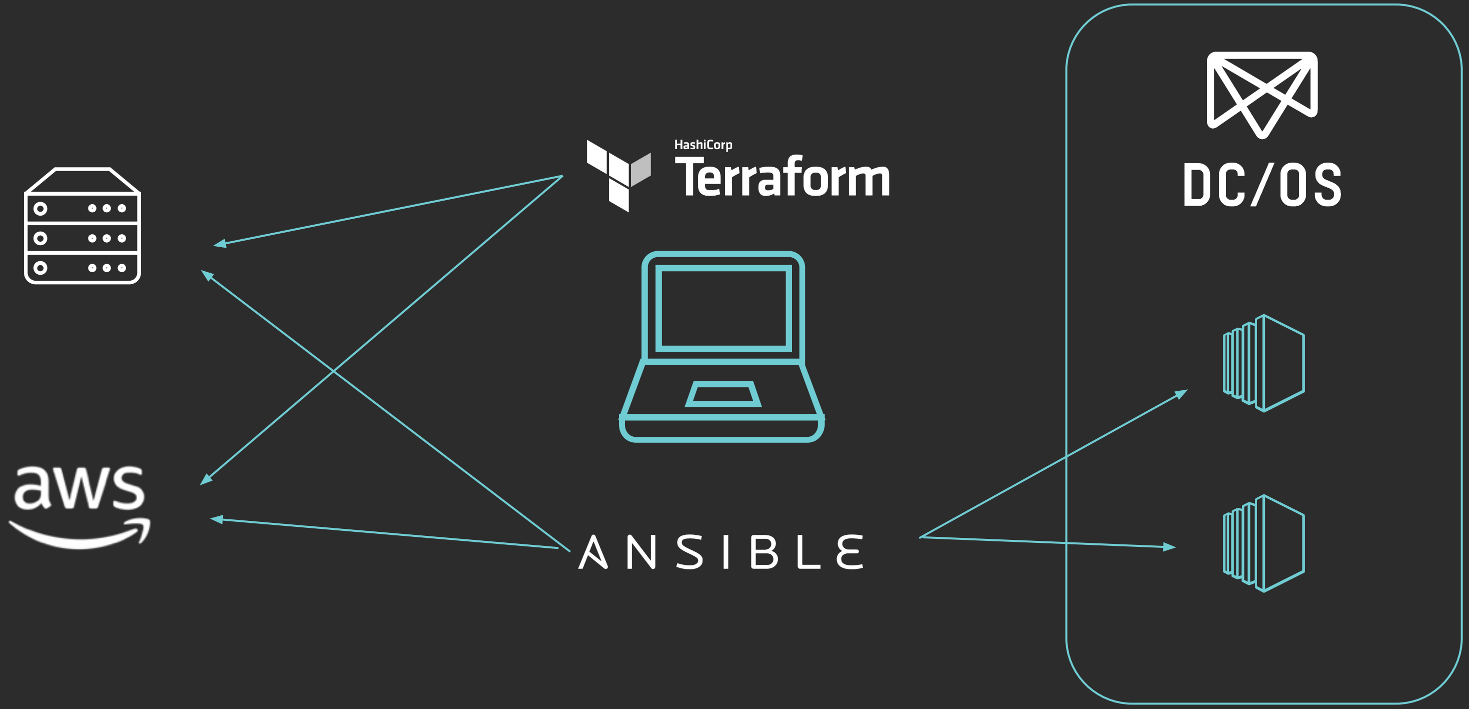
Host Instance

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

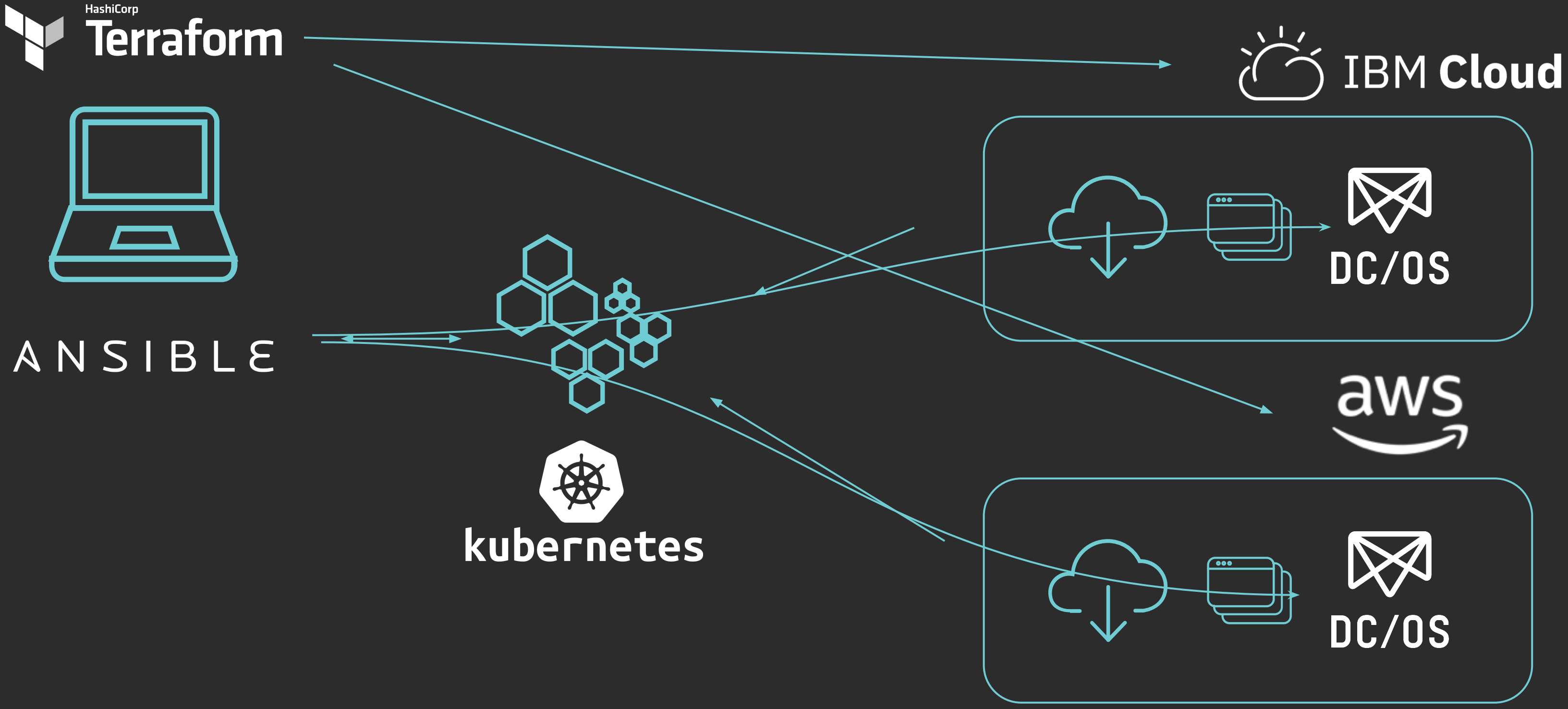
Patterns

Jonathan Owens

Old Management Architecture



Kubernetes API Inventory



Thank You

Jonathan Owens | @intjonathan | jonathan@newrelic.com

Maryum Styles | mstyles@newrelic.com

New Relic.