The World is Blue / Green

GIDS 2018
दुनिया नीली और हरा है
Ken Sipe
Distributed Systems Engineer

@kensipe
THE WORLD HAS CHANGED AND WE ARE ALL NOW SOFTWARE COMPANIES.

Software is eating the world.

Marc Andreessen
What do you need for Blue / Green?
What Datacenter long long ago…
Challenge: Static Partitioning
Today’s Legacy Datacenter

Provision VMs in the cloud or on physical servers

mesosphere
Installing an Application with Static Partitioning

Install Hadoop on a static set of machines
Installing an Application with Static Partitioning

Install Web Server on a static set of machines
Resizing an Application with Static Partitioning

Scale up Hadoop manually
What if your Laptop was operated like your Data Center?
Ops / DevOps Tools
In a modern datacenter...
Micro - Services
Orchestration
MESOS LETS YOU TREAT A CLOSER OF NODES…
AS ONE BIG COMPUTER
DATACENTER OPERATING SYSTEM
"An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs."
“An operating system (OS) is system software that manages datacenter and software resources and provides common services for computer programs.”
The UNIX Operating System Stack

Applications
- Apache
- MySQL
- Memcached
- SSHd

Init System
- Init, Upstart, Systemd

Kernel
- Linux, BSD

mesosphere
The Mesosphere Operating System Stack

- Rails
- Redis
- Elasticsearch
- Memcached

- Marathon

- Mesos

Applications
Init System
Kernel
DCOS STACK

Any Service or Container
Your favorite services, container formats, and those yet to come

Mesosphere DCOS
Runs distributed apps anywhere as simply as running apps on your laptop

Any Infrastructure
Build apps once on DCOS, and run it anywhere
DCOS NODE TYPES
DC/OS Architecture Overview

Services & Containers

- HDFS
- Jenkins
- Marathon
- Cassandra
- Flink
- Spark
- Docker
- Kafka
- MongoDB
- +30 more...

DC/OS

- Container Orchestration
- Security & Governance
- Monitoring & Operations
- User Interface & Command Line

ANY INFRASTRUCTURE

- Physical Servers
- Virtual Servers
- Private Cloud
- Public Cloud Providers (Google, AWS, Azure)
tobi@Zwerg ~/dcos$ dcos help

Command line utility for the Mesosphere Datacenter Operating System (DCOS). The Mesosphere DCOS is a distributed operating system built around Apache Mesos. This utility provides tools for easy management of a DCOS installation.

Available DCOS commands:

- **config**  Get and set DCOS CLI configuration properties
- **help**    Display command line usage information
- **kafka**   Start and manage Kafka brokers
- **marathon** Deploy and manage applications on the DCOS
- **node**    Manage DCOS nodes
- **package** Install and manage DCOS packages
- **service** Manage DCOS services
- **spark**   Run and manage Spark jobs
- **task**    Manage DCOS tasks

Get detailed command description with 'dcos <command> --help'.

tobi@Zwerg ~/dcos$
Blue/Green Upgrades

- Two separate services are created, each of which is running a different version of your application
  - One deployment is currently production
  - One deployment will be production
- Access to what is currently considered production is done through an external load balancer
  - In the context of DC/OS, the load balancer can be provided by either Marathon-LB or Edge-LB
- After the to-be production service has been launched and tested by an administrator, the load balancer is re-configured to direct traffic to the new version
- Old version can be kept around in case a flip-over is required due to failures in the new version
- Process repeats when the next version is ready to be rolled out
Blue/Green Upgrades

App Client/User

Load Balancer

instance-1
state=R
hc=N/A

instance-2
state=R
hc=N/A

instance-3
state=R
hc=N/A

instance-4
state=R
hc=N/A

v1.0
Blue/Green Upgrades

App Client/User

Load Balancer

v1.0

instance-1
state=R
hc=N/A

instance-2
state=R
hc=N/A

instance-3
state=R
hc=N/A

instance-4
state=R
hc=N/A

v1.1

instance-1
state=S
hc=N/A

instance-2
state=S
hc=N/A

instance-3
state=S
hc=N/A

instance-4
state=S
hc=N/A

Rolling Upgrades

DC/OS Fundamentals
Blue/Green Upgrades

App Client/User

Load Balancer

instance-1
state=R
hc=N/A

instance-2
state=R
hc=N/A

instance-3
state=R
hc=N/A

instance-4
state=R
hc=N/A

v1.0

instance-1
state=R
hc=N/A

instance-2
state=R
hc=N/A

instance-3
state=R
hc=N/A

instance-4
state=R
hc=N/A

v1.1
Blue/Green Upgrades

App Client/User

Load Balancer

v1.0
instance-1
state=R
hc=N/A
instance-2
state=R
hc=N/A
instance-3
state=R
hc=N/A
instance-4
state=R
hc=N/A

v1.1
instance-1
state=R
hc=H
instance-2
state=R
hc=H
instance-3
state=R
hc=H
instance-4
state=R
hc=H
Blue/Green Upgrades

App Client/User

Load Balancer

DC/OS Operator/Admin

reconfigures

instance-1
state=R
hc=N/A

instance-2
state=R
hc=N/A

instance-3
state=R
hc=N/A

instance-4
state=R
hc=N/A

v1.0

instance-1
state=R
hc=H

instance-2
state=R
hc=H

instance-3
state=R
hc=H

instance-4
state=R
hc=H

v1.1
Blue/Green Upgrades

App Client/User

Load Balancer

v1.0

instance-1
state=R
hc=N/A

instance-2
state=R
hc=N/A

instance-3
state=R
hc=N/A

instance-4
state=R
hc=N/A

v1.1

instance-1
state=R
hc=H

instance-2
state=R
hc=H

instance-3
state=R
hc=H

instance-4
state=R
hc=H
Blue/Green Upgrades
RELIABLE, SIMPLIFIED CI/CD INTEGRATION

Continuous Delivery Pipeline

Version Control System → Continuous Integration → Container Orchestrator → Production Environment → Load Balancer

Artifact Repo & Container Registry

git push
RELIABLE, SIMPLIFIED CI/CD INTEGRATION with DC/OS

Continuous Delivery Pipeline

GitLab, Bitbucket, GitHub

Jenkins

Marathon

DC/OS (Mesos)

Marathon-lb

Artifactory, Nexus

Apache Mesos & DC/OS

git push
APPLICATION LIFECYCLE

Build and deploy modern apps on the same infra

DC/OS

- Identical Infrastructure across Test/Staging/Production with strong isolation
- Self service access

BENEFITS

Less developer time troubleshooting environment issues

Easy experimentation with new technologies
**ZERO-DOWNTIME DEPLOYMENTS**

### Rolling Deployment

Bring up new version and terminate old ones until all old are gone.

- T0: [0.9] [0.9] [0.9] [0.9]
- T1: deployment kicks off
- T2: [0.9] [0.9] [0.9] [0.9] [1.0]
- T3: [0.9] [0.9] [1.0] [1.0] [1.0]
- T4: [0.9] [1.0] [1.0] [1.0] [1.0]
- T5: [1.0] [1.0] [1.0] [1.0] [1.0]
- T6: deployment done

### Blue-Green Deployment

Launch a new stack and switch traffic from old to new when new instances are healthy.

### Canary Deployment

Bring up a new version, start by routing a *small portion* of traffic to the new app, and slowly increase.
# DEPLOYING APPS

## Manual

**Scheduling**
- A sysadmin provisions one or more physical/virtual servers to host the app

**Deployment**
- By hand or using Puppet / Chef / Ansible
- Jenkins SSHing to the machine and running a shell script
- Note: all dependencies must also be present!

**Health checks**
- Nagios pages a sysadmin

**Service discovery**
- Static hostnames / IP addresses in a spreadsheet or config management
- A sysadmin configures a load balancer manually or with Puppet / Chef / Ansible

**Persistence**
- Individual servers with RAID 1/5/6/10, expensive SANs, NFS, etc.
- Dedicated, statically partitioned Ceph or Gluster storage clusters

## Automatic

**Scheduling**
- Mesos resource offers (two-tier scheduling) offers available resources directly to frameworks

**Deployment**
- Containers deployed, ideally using a CI/CD tool to create/update app definitions
- Docker containers packages app and dependencies

**Health checks**
- Health checks, restarts unhealthy/failed instances

**Service discovery**
- Provides DNS resolution for running services (hostname / IP address, ports, etc)
- Load balancer configs built dynamically using cluster state

**Persistence**
- External/persistent volumes (REX-Ray), HDFS, etc.
- Self-healing Ceph or Gluster on Mesos / DC/OS
CASE STUDY: STRAVA DEPLOY PROCESS

Pre-Container
- Build a Debian package
- Push deb pkg to apt server
- Wait for apt server to have deb package ready
- Boot a new AWS instance
- Run puppet on the instance, installing the deb
- Turn the instance into an AMI image
- Boot new AWS instances using new AMI
- Terminate old AWS instances

Container
- Build and push Docker image
- Deploy

It would easily take 30 minutes for a single deploy even under ideal conditions where nothing broke.

A simple service might only take 20 seconds to fully deploy under ideal conditions.
CASE STUDY: Spotify
References:

- [https://dcos.io/](https://dcos.io/)
- [https://vamp.io/documentation/installation/v0.9.4/dcos/](https://vamp.io/documentation/installation/v0.9.4/dcos/)
- [https://github.com/kensipe/Clients](https://github.com/kensipe/Clients)

Thanks to:
- Michael Hausenblas @mhausenblas
- Jörg Schad @joerg_schad
- Ju Stroh @ju_stroh
ANY QUESTIONS?

@kensipe
Join the DC/OS Community

Connect with our community of users and browse the latest DC/OS news.

GitHub
Are you interested in helping us make DC/OS even better? Let’s work together! Check out our source code on GitHub.
View repositories →

Slack
Have any questions? Our Slack channel is the best place to get help. Just send us a request to automatically receive your invitation.
Join chat →

Mailing List
Want to stay in the loop and connect with other community members? Our public mailing list has all the latest updates. Join the discussion.
Join users@dcos.io →
GREAT INDIAN DEVELOPER SUMMIT 2019
Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com  @greatindiandev  bit.ly/gidslinkedin