SOFTWARE ARCHITECTURE | NOVEMBER 15, 2016

CONTINUOUS DELIVERY WITH DC/OS AND JENKINS

MESOSPHERE

AGENDA

Presentation

- Introduction to Apache Mesos and DC/OS
- Components that make up modern infrastructure
- Running Jenkins as a service on DC/OS
- Continuously deploying applications to DC/OS

Demos & Lab

- Installing and configuring Jenkins
- Installing and configuring a load balancer
- Creating a new CI/CD pipeline
- Putting it all together (CD in practice)

DEVELOPER AGILITY, DEFINED

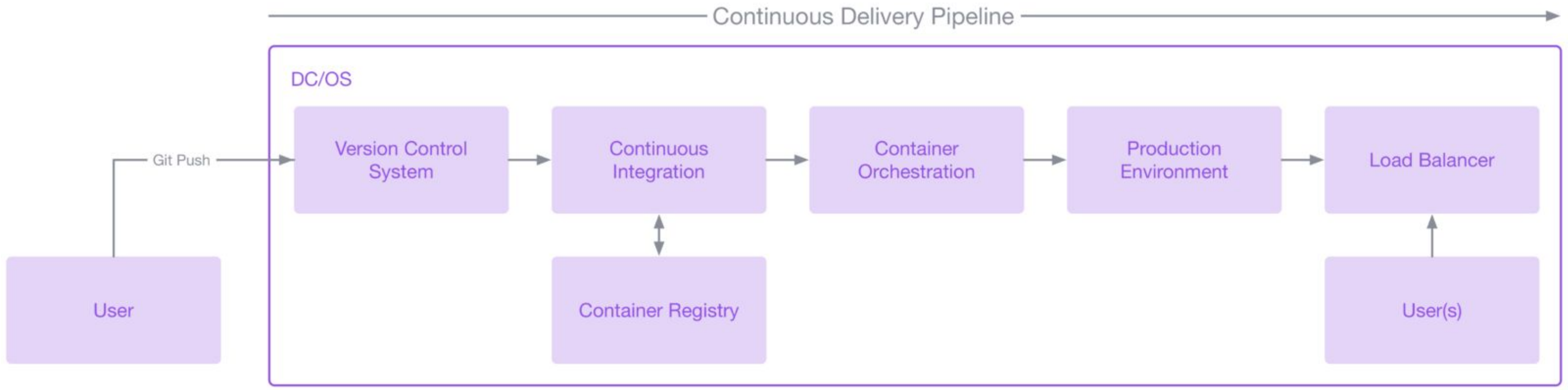
DEVELOPER AGILITY, DEFINED

Developer agility empowers developers to

- ship their apps to production
- leverage the power of Mesos and DC/OS
- fix bugs rapidly

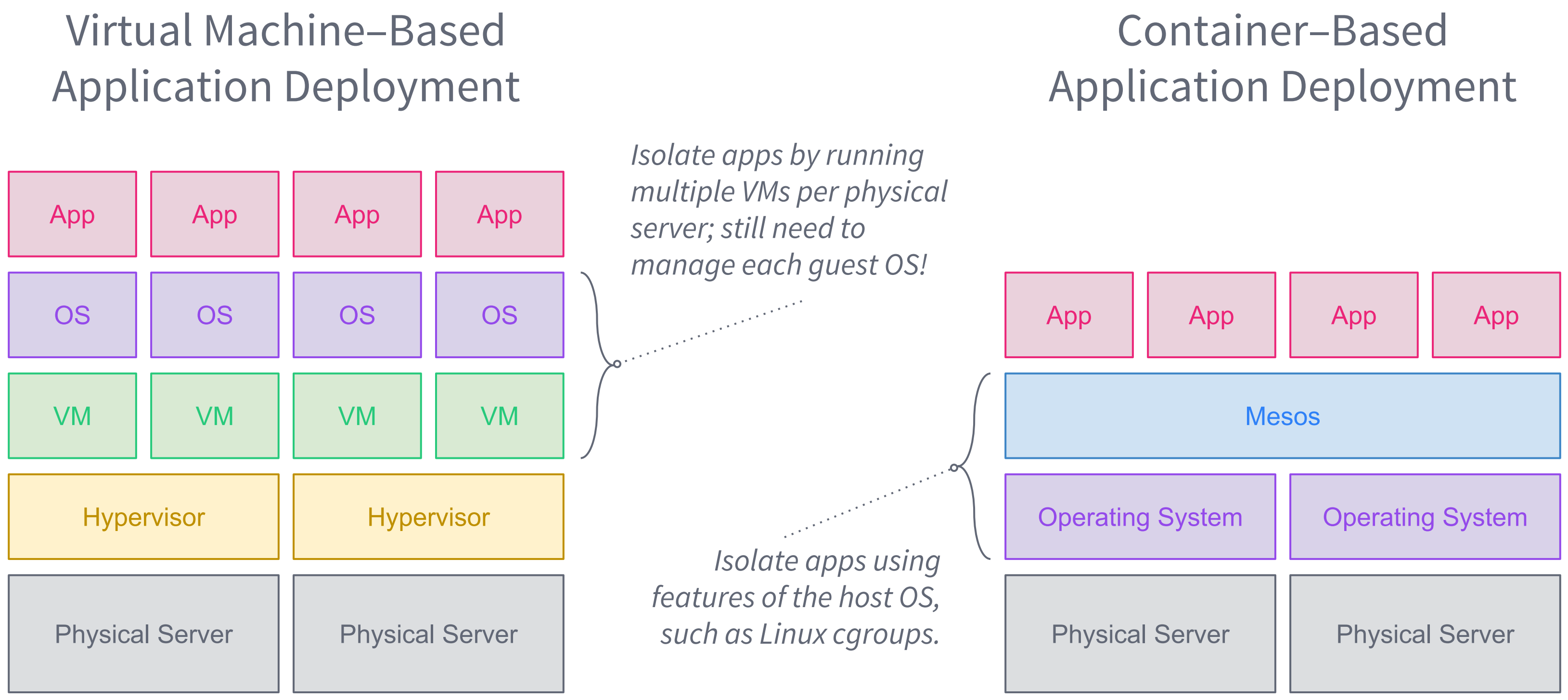
without downtime!

DEVELOPER AGILITY, DEFINED

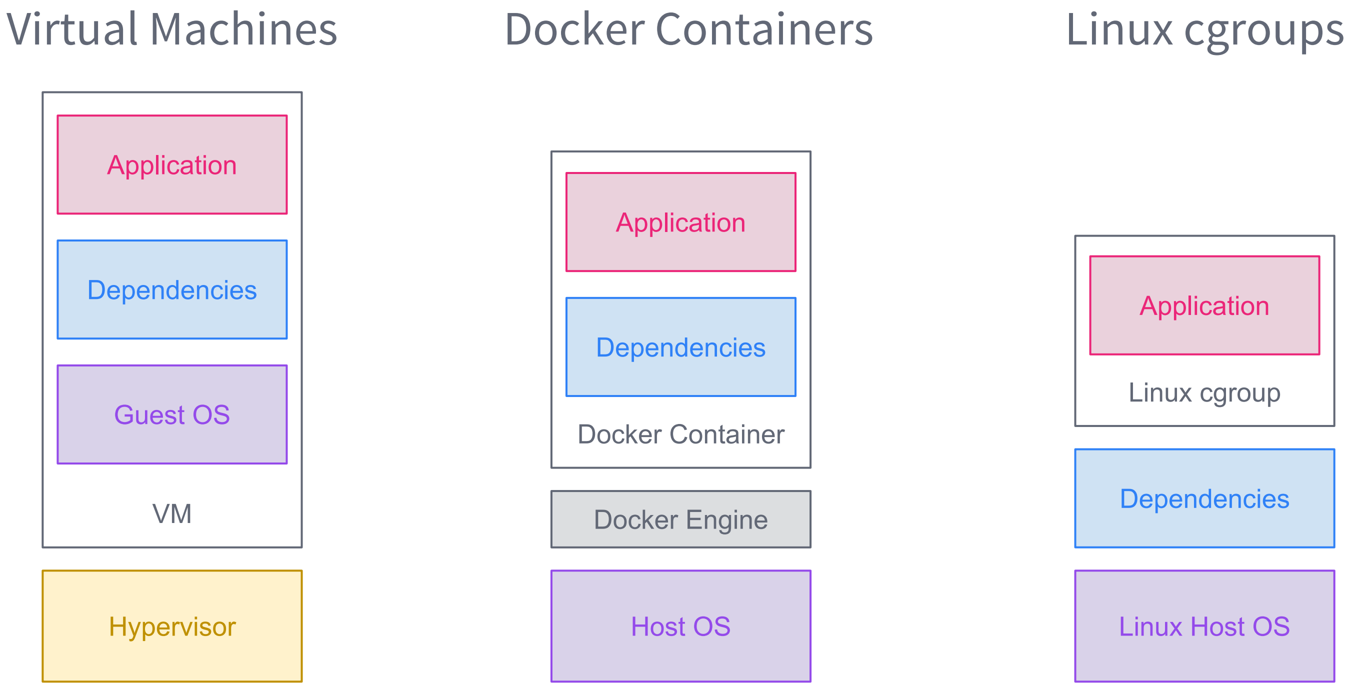


INTRO TO APACHE MESOS AND DC/OS

A QUICK PRIMER ON CONTAINERS



A QUICK PRIMER ON CONTAINERS



A BIT OF CLARIFICATION

Apache MESOS™

DC/OS

https://mesos.apache.org

https://dcos.io

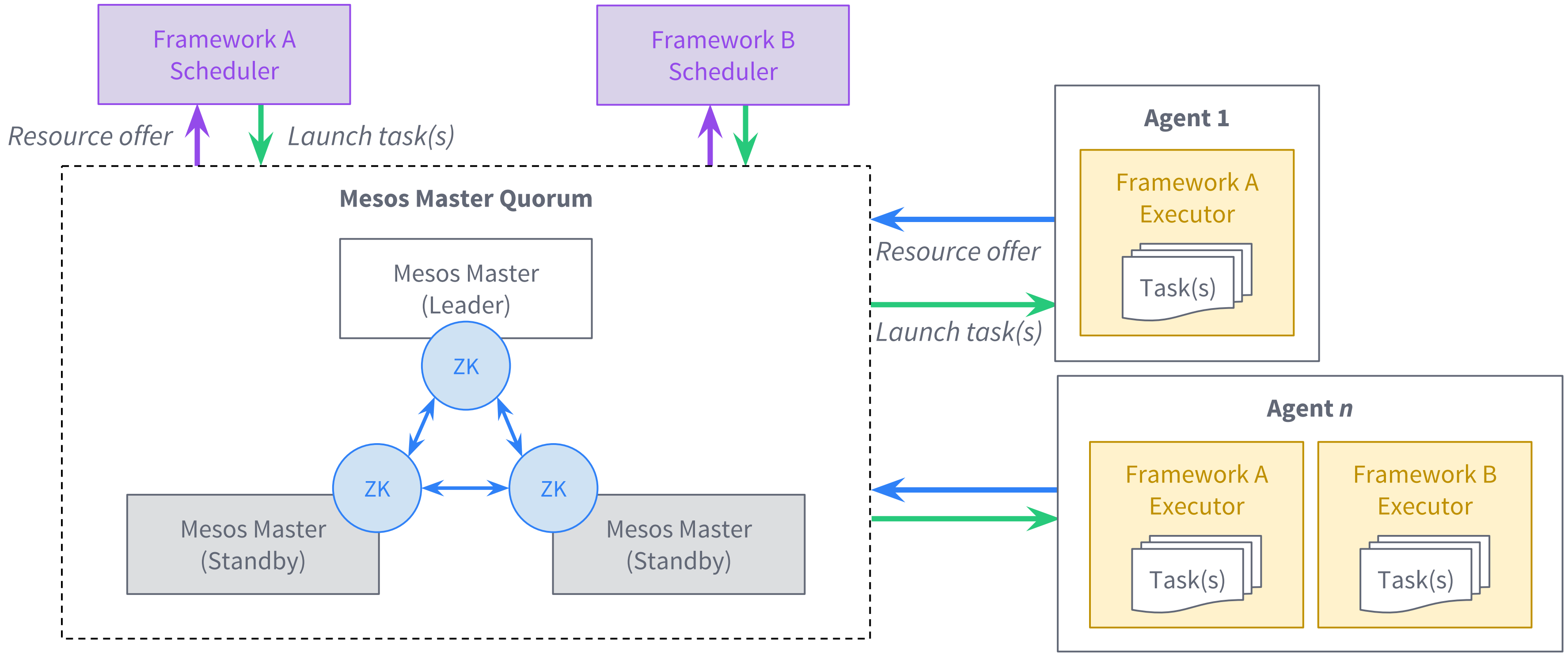
WHAT IS MESOS?

- General purpose cluster resource manager
- Represents many machines as a single entity
- Advertises resources directly to *frameworks*
- Works at scale: Apple, Twitter, Airbnb, Netflix, …

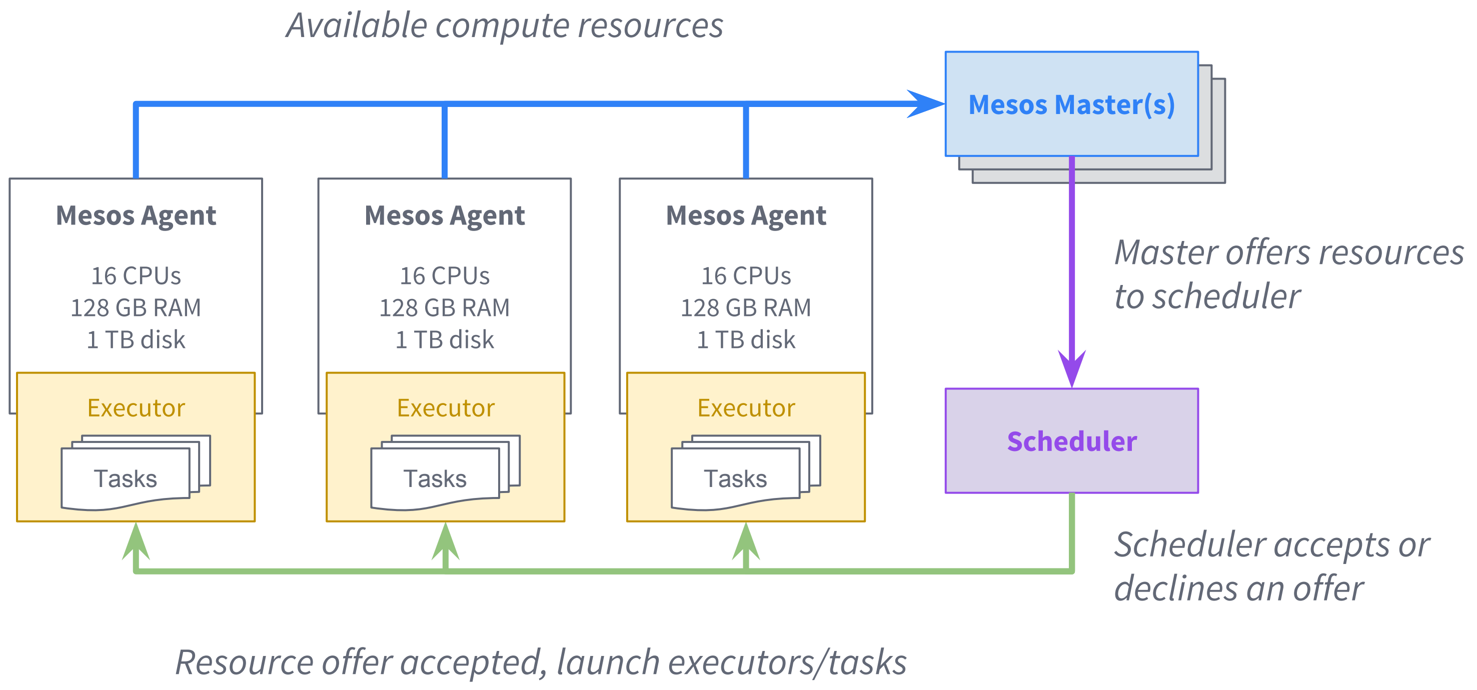
WHAT IS MESOS? (CONTINUED)

- Two-tier scheduling across resource types
    - cpus, mem, disk, and ports by default
- Masters are highly available, agents are fault tolerant
    - *Checkpointing*, *agent recovery*
- Resource isolation between processes
    - Linux cgroups, Docker, …
- Language bindings: C++, Java, Python, Go, …

MESOS ARCHITECTURE



ANATOMY OF A RESOURCE OFFER (TWO-TIER SCHEDULING)



NEW (OLD) PROBLEMS

- Service discovery and load balancing
    - BIND, Mesos-DNS, Consul-Mesos, Marathon-LB

NEW (OLD) PROBLEMS

- Service discovery and load balancing
  - BIND, Mesos-DNS, Consul-Mesos, Marathon-LB
- Monitoring and metrics collection
  - Collectd, Nagios, Prometheus, Snap

NEW (OLD) PROBLEMS

- Service discovery and load balancing
  - BIND, Mesos-DNS, Consul-Mesos, Marathon-LB
- Monitoring and metrics collection
  - Collectd, Nagios, Prometheus, Snap
- Persistent storage (filesystems, databases, etc)
  - Ceph, HDFS, Amazon EBS / EFS / S3, NFS, Cassandra

NEW (OLD) PROBLEMS

- Service discovery and load balancing
  - BIND, Mesos-DNS, Consul-Mesos, Marathon-LB
- Monitoring and metrics collection
  - Collectd, Nagios, Prometheus, Snap
- Persistent storage (filesystems, databases, etc)
  - Ceph, HDFS, Amazon EBS / EFS / S3, NFS, Cassandra
- Administration: named URIs vs. ports, IPAM
  - Nginx, HAProxy, Mesos-DNS, dhcpd, Minuteman

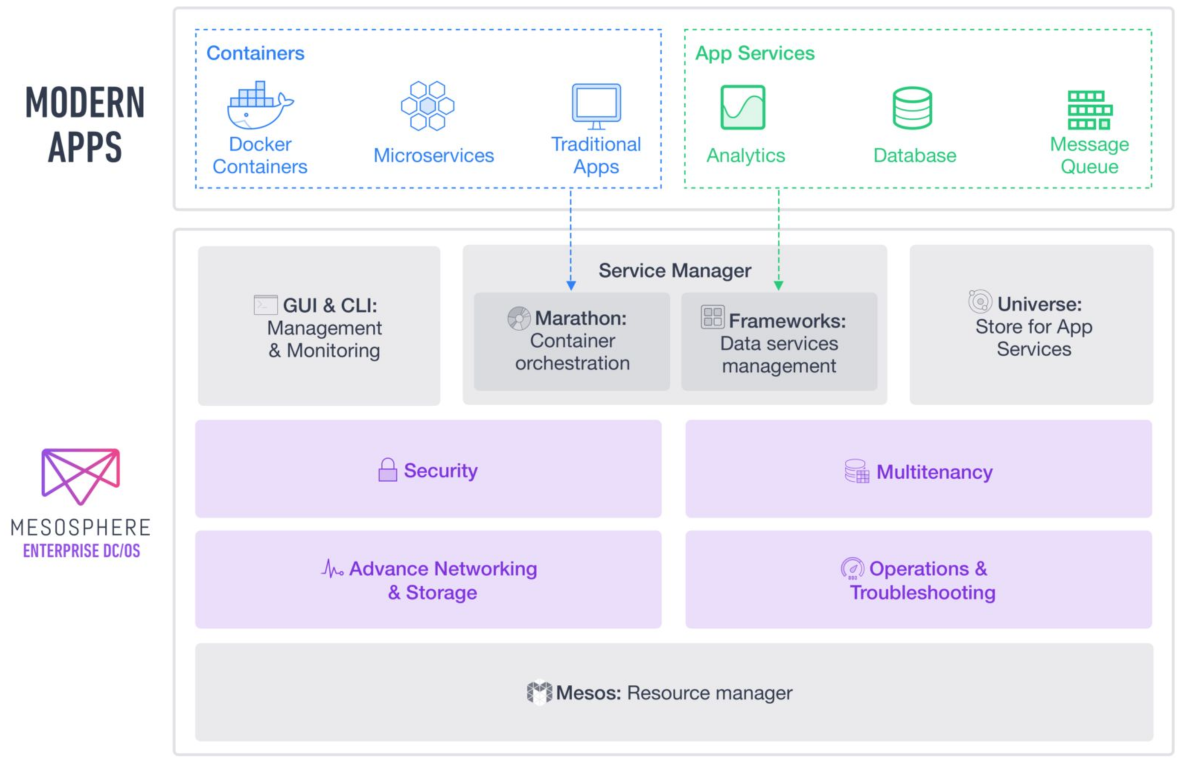
DC/OS: BUILT ON MESOS

DC/OS

https://dcos.io

https://github.com/dcos

DC/OS: BUILT ON MESOS


MODERN APPS
Containers
Docker Containers
Microservices
Traditional Apps
App Services
Analytics
Database
Message Queue
Service Manager
GUI & CLI: Management & Monitoring
Marathon: Container orchestration
Frameworks: Data services management
Universe: Store for App Services
MESOSPHERE ENTERPRISE DC/OS
Security
Multitenancy
Advance Networking & Storage
Operations & Troubleshooting
Mesos: Resource manager

MESOS AND DC/OS: BETTER TOGETHER

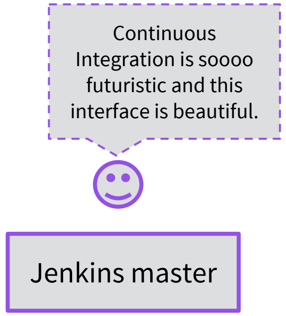
All of the benefits of Mesos, plus

- Built-in service discovery and load balancing
- Support for stateful services
- Turn-key installation of distributed systems
- Cloud-agnostic installer
- Web and command-line interfaces
- All components are integration tested and supported by Mesosphere, Inc.

JENKINS ON DC/OS

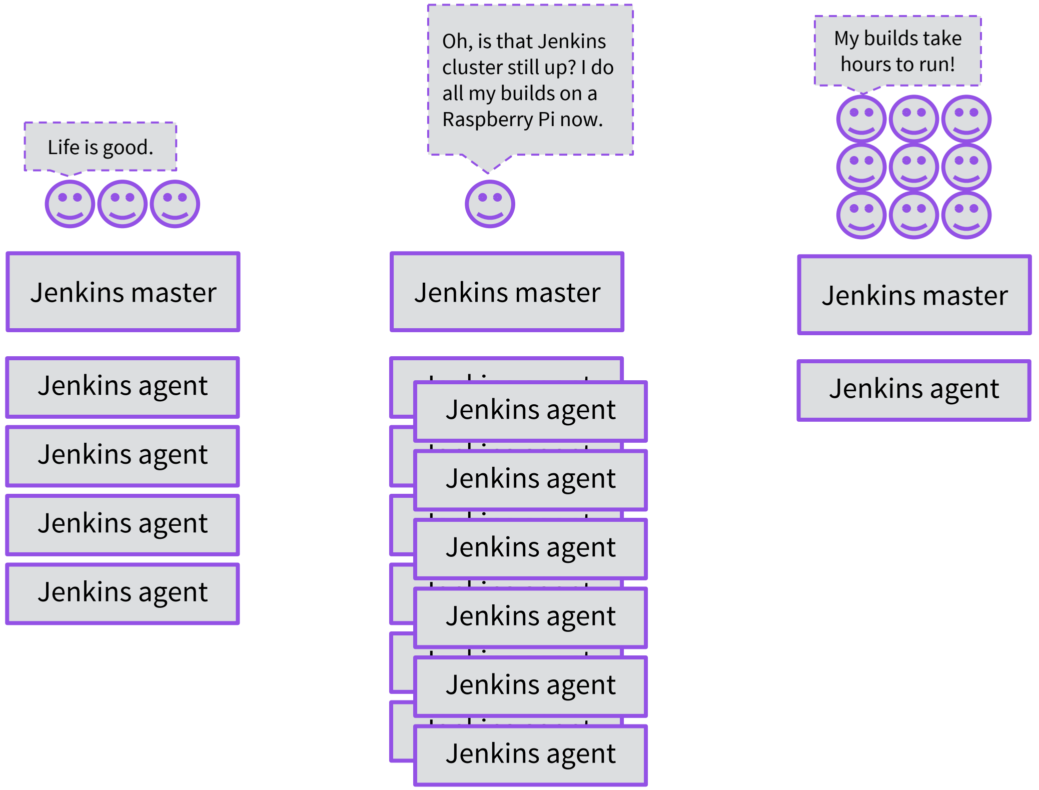
Jenkins on DC/OS

WHEN IT BEGAN


Continuous Integration is soooo futuristic and this interface is beautiful.
Jenkins master

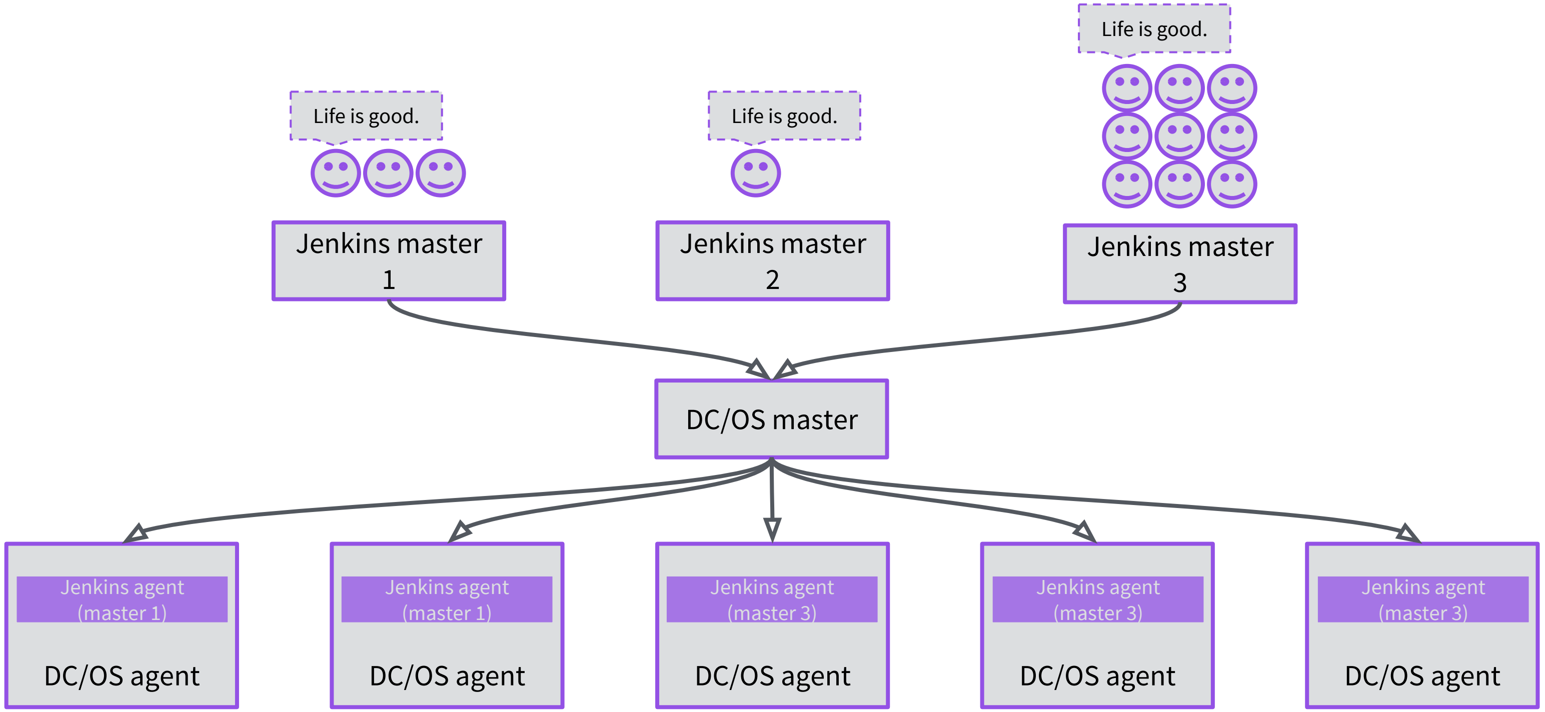
Jenkins on DC/OS

THE OLD WORLD



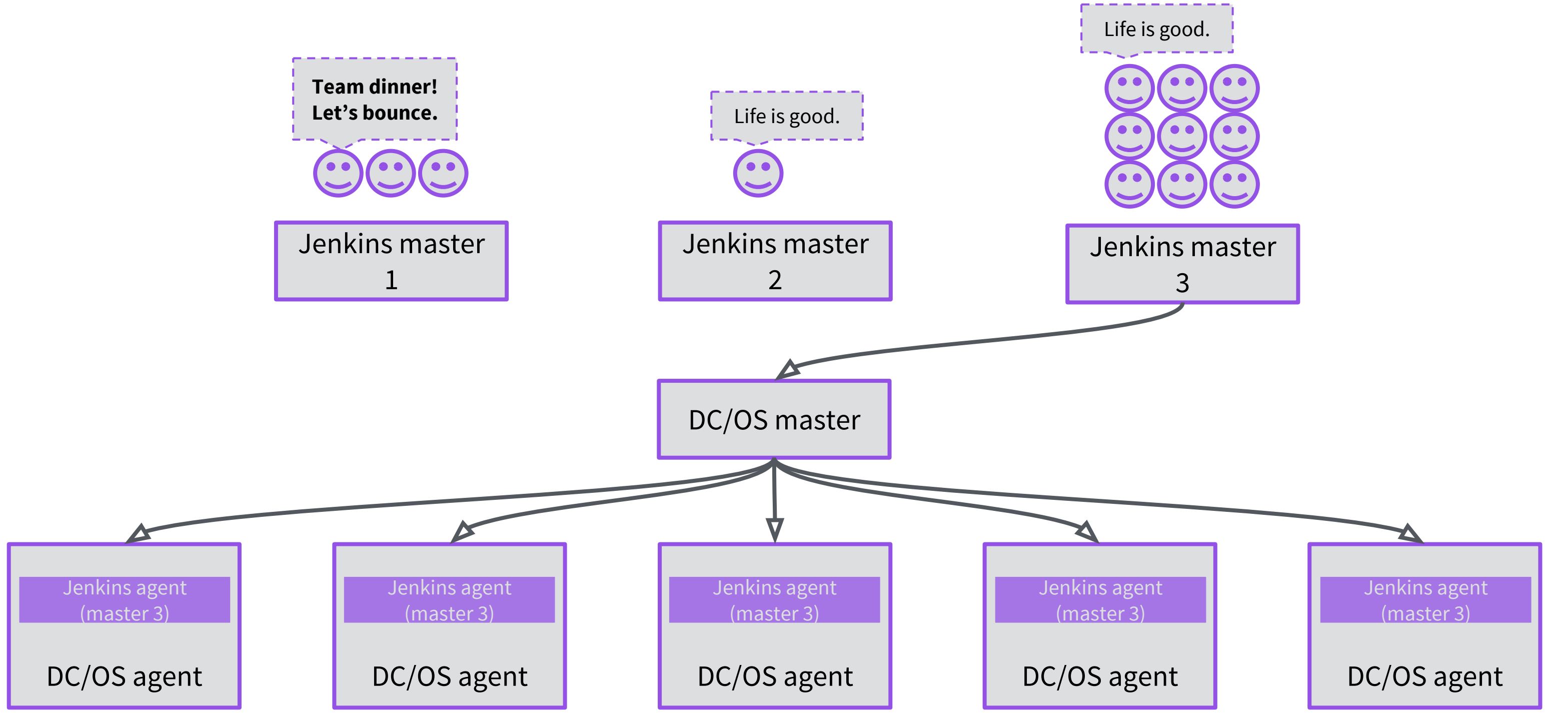
Jenkins on DC/OS

RESOURCE EFFICIENCY



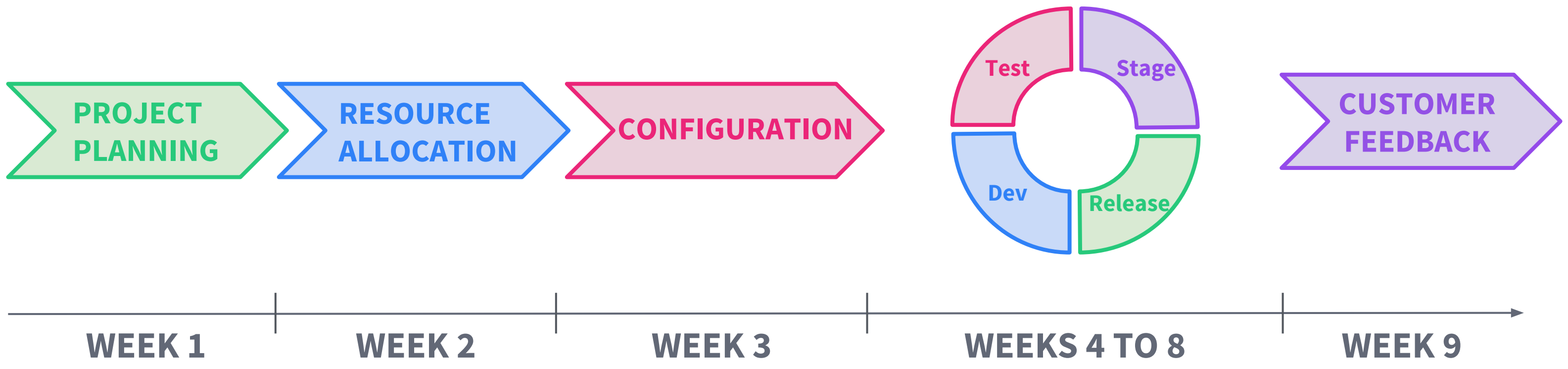
Jenkins on DC/OS

RESOURCE EFFICIENCY



CONTINUOUSLY DEPLOYING APPLICATIONS TO DC/OS

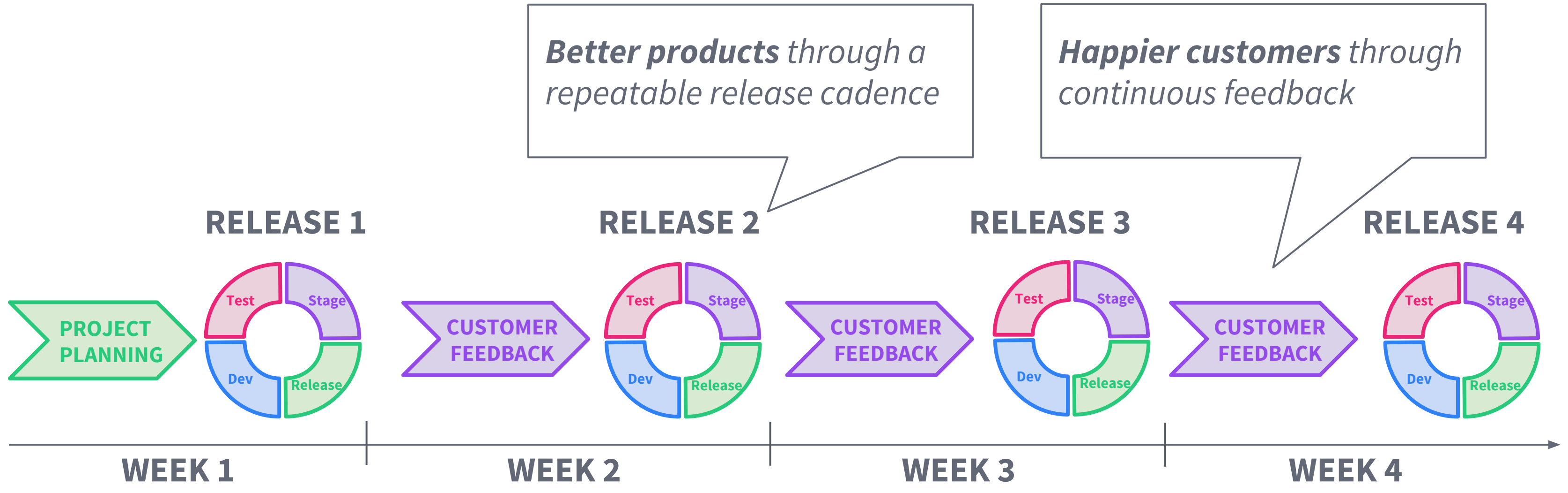
TRADITIONAL RELEASE PROCESS



DEV(OPS) TEAMS SPEND SIGNIFICANT TIME AND EFFORT ON:

- Planning & implementing new technologies
- Waiting for people & infrastructure
- Building environment specific CI/CD for each project
- Moving apps from dev to staging to prod

MODERN RELEASE PROCESS



DEPLOYING APPLICATIONS: BASIC REQUIREMENTS

- ***Scheduling*** — advertising available compute resources
- ***Deployments*** — getting an application onto a node
- ***Health checks*** — ensuring the app/service is healthy
- ***Service discovery*** — connecting to dependent services
- ***Persistence*** — running stateful services in containers

DEPLOYING APPLICATIONS: SCHEDULING

Before DC/OS	With DC/OS
A sysadmin provisions one or more physical/virtual servers to host the app	Mesos resource offers (two-tier scheduling) offers available resources directly to frameworks

DEPLOYING APPLICATIONS: DEPLOYMENTS

Before DC/OS

By hand or using Puppet / Chef / Ansible

Jenkins SSHing to the machine and running a shell script

Note: all dependencies must also be present!

With DC/OS

Marathon deploys containers, ideally using a CI/CD tool to create/update app definitions

Docker containers packages app and dependencies

DEPLOYING APPLICATIONS: HEALTH CHECKS

Before DC/OS	With DC/OS
Nagios pages a sysadmin	Marathon performs health checks, restarts unhealthy/failed instances

DEPLOYING APPLICATIONS: SERVICE DISCOVERY

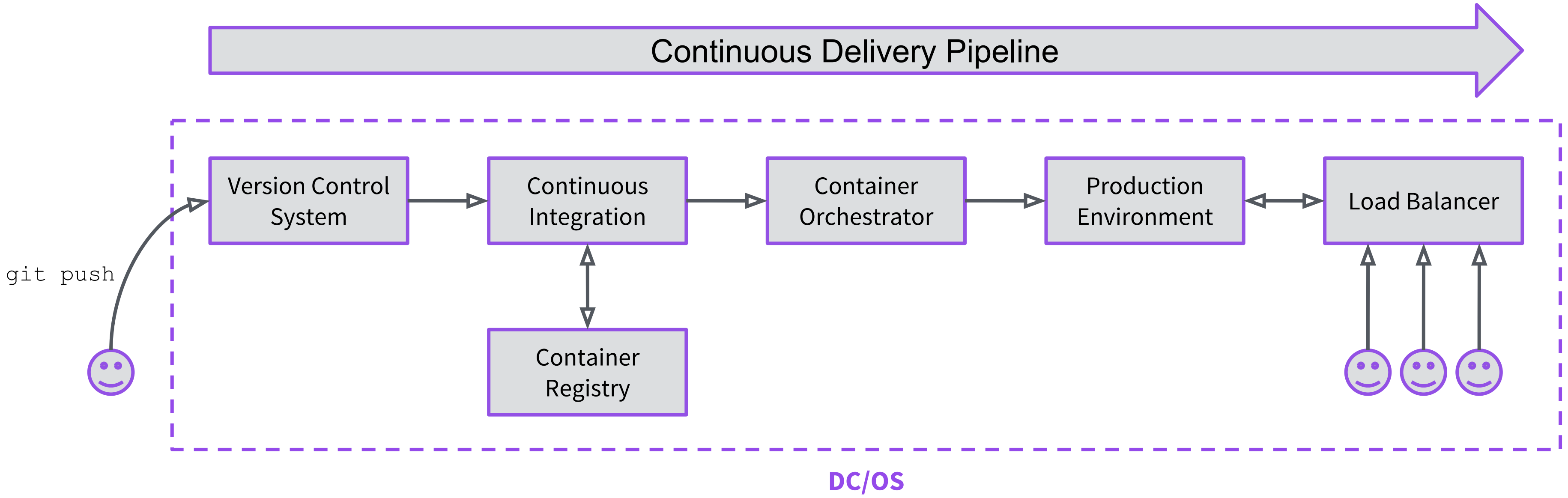
Before DC/OS	With DC/OS
Static hostnames / IP addresses in a spreadsheet or config management A sysadmin configures a load balancer manually or with Puppet / Chef / Ansible	Mesos-DNS provides DNS resolution for running services (hostname / IP address, ports, etc) Load balancer configs built dynamically using cluster state

DEPLOYING APPLICATIONS: PERSISTENCE

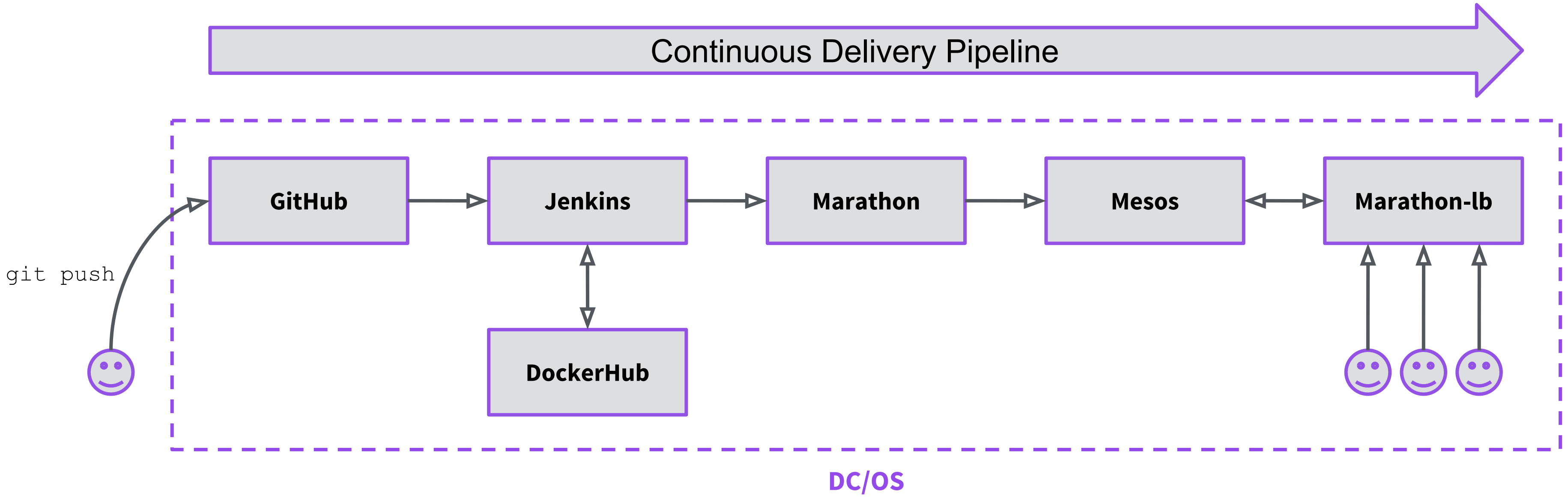
Before DC/OS	With DC/OS
Individual servers with RAID 1/5/6/10, expensive SANs, NFS, etc.	Mesos external/persistent volumes (REX-Ray), HDFS, etc.
Dedicated, statically partitioned Ceph or Gluster storage clusters	Self-healing Ceph or Gluster on Mesos / DC/OS

DEMOS & LAB

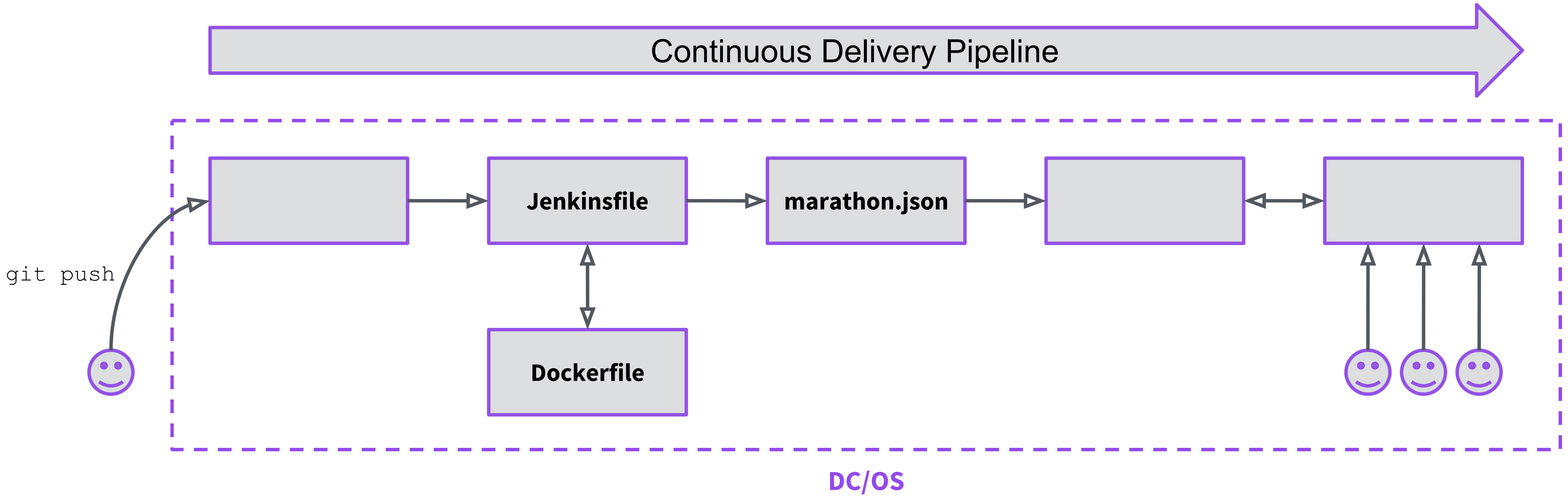
PIPELINE COMPONENTS



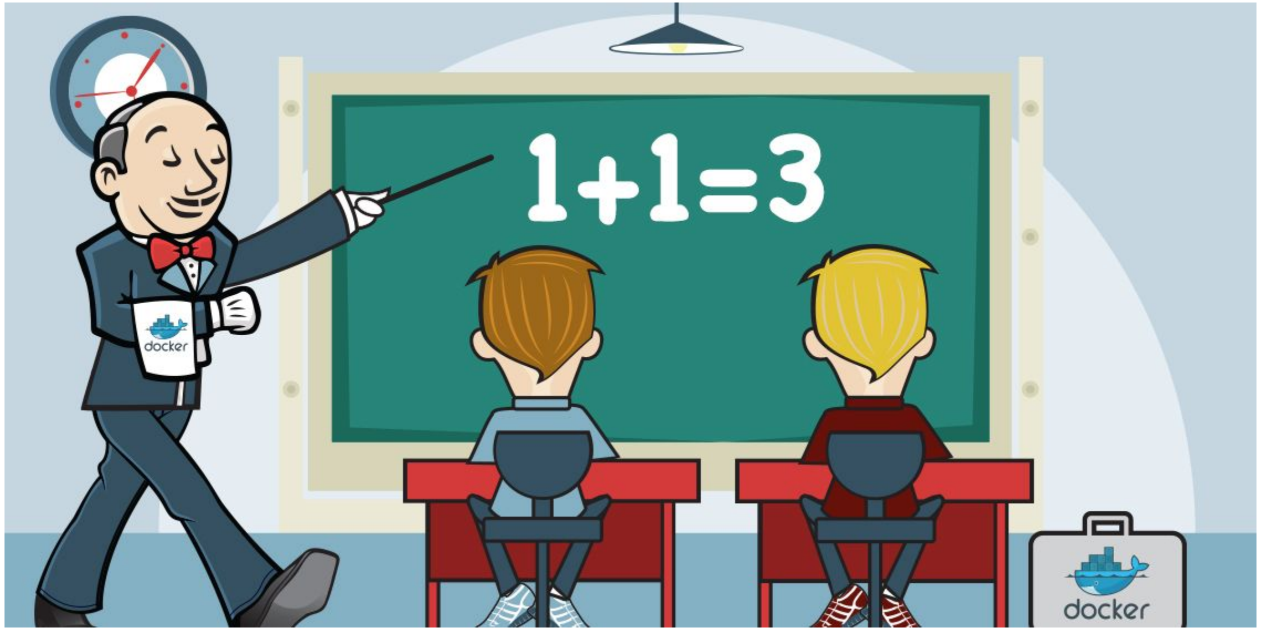
PIPELINE COMPONENTS



PIPELINE CONFIGURATION



A SNEAK PREVIEW



THANK YOU!

Sunil Shah
sunil@mesosphere.com
@ssk2

Learn more by visiting dcos.io and mesosphere.com