DC/OS Day Two Operations

Day One

MESOSPHERE

Course Goals

1. Learn about backup and restore operations
2. Understand the upgrade process
3. Learn how to access DC/OS logs
4. Integrate DC/OS with an external log aggregation system
5. Measure performance and health metrics
6. Discuss failure and recovery scenarios

Course Prerequisites

Experience:

- Attended DC/OS Fundamentals or equivalent experience
- Basic understanding of containerization
- Intermediate Linux background

Required software:

- SSH client
- Web browser (Firefox or Chrome)

Classroom Lab Environment

Your instructor will provide 5 hosts running on AWS with a list of IP addresses that identify how to connect to your cluster nodes.

The list will include:

- 5 publicly accessible IP addresses along with internal IP addresses in the 10.0.0.x subnet
- 1 bootstrap node, 1 master node, 1 public agent, and 2 private agents

Agenda:

Day One

1. DC/OS Architecture Review
2. Backup and Restore with dcos
3. Rolling Upgrades
4. Log Management

Agenda:

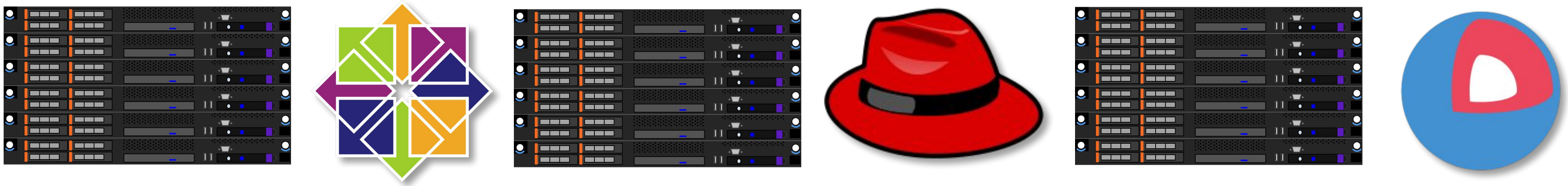
Day Two

1. Monitoring and Metrics
2. Failure Handling
3. Production Checklist

DC/OS Architecture Review

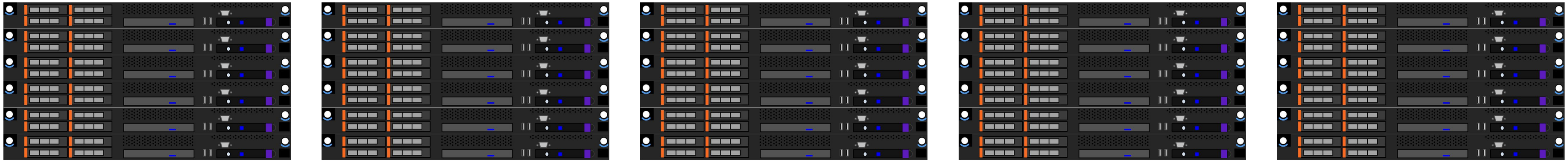
Cluster Building Blocks

- Clusters are made up of Linux servers where each one is referred to as a node
- DC/OS 1.11 supports the following operating systems:
  - Red Hat Enterprise Linux 7.4
  - CentOS 7.4
  - CoreOS 1235.12.0
- Different nodes have different functions/responsibilities and as such have different hardware requirements and recommendations



Types of Nodes

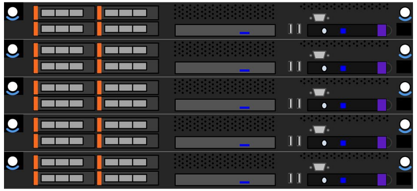
- DC/OS clusters are made up of two main types of nodes: **master** nodes and **agent** nodes
- **Masters** run the cluster, run many APIs, store the state of the cluster, and provide an entry point for a request into the cluster
- **Agents** are often referred to as **slave** nodes in both config files, documentation and in both CLI and GUI output - they are the systems in the cluster which provide resources to allocate to your workloads



Master Nodes - Hardware

- Master nodes control the Mesos layer and run many key processes for DC/OS
- Must come in odd numbers for Zookeeper election purposes
  - One: Training, testing, proof of concept
  - Three: Development
  - Five: Development or Production
- Masters do not horizontally scale - you choose your number before you install
- Base hardware requirements (see right) are sufficient for small clusters: < 20 nodes
- Size up memory/CPU for larger clusters
- Consider SSDs regardless of cluster size - Zookeeper loves them

Master Nodes

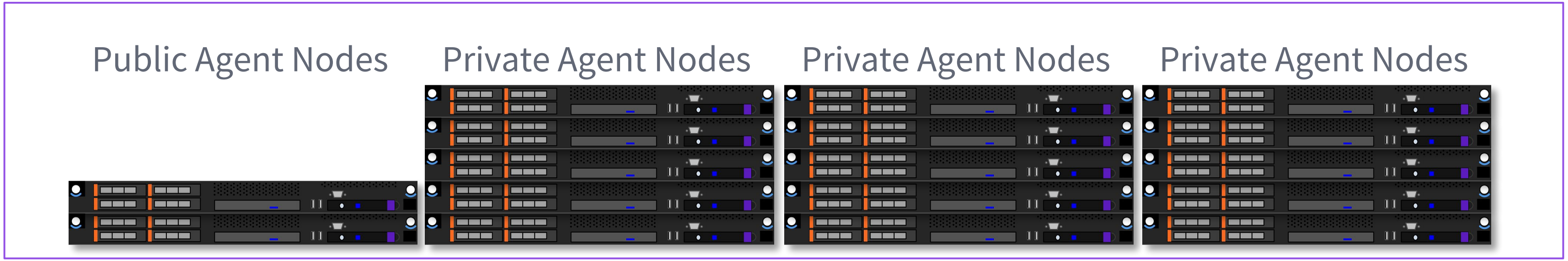


	Minimum	Recommended
RHEL/CentOS	7.4	7.4
CoreOS	1235.12.0	1235.12.0
Nodes	1	3 or 5
CPU	4 cores	4 cores
RAM	32 GB	32 GB
Disk	120 GB	120 GB

Agent Nodes - Hardware

- Agent nodes run your applications
- Agent nodes also run some DC/OS related processes
- Agents can be added to an existing cluster at any time
- Number of agents along with their sizing depends greatly on the requirements of your applications

	Recommended
RHEL/CentOS	7.4
CoreOS	1235.12.0
Nodes	6 or more
CPU	2 cores
RAM	16 GB
Disk	60 GB

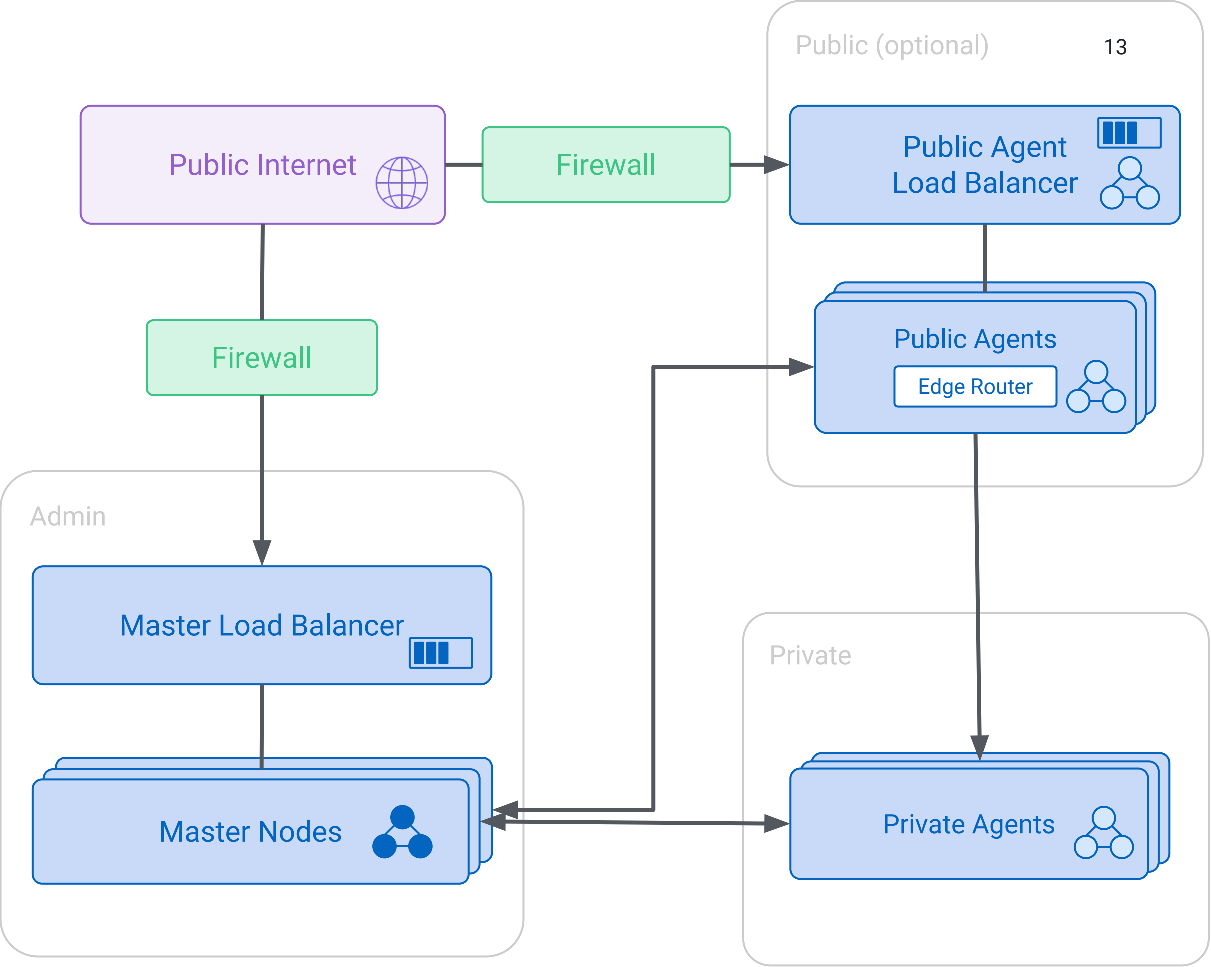


Agent Nodes: Public vs. Private

There are two types of agent nodes: **public** agents and **private** agents

Public Agents	Private Agents
Run services that facilitate connections outside the cluster - usually a load balancer	The default agent and run most everything else
Exist on a network that is accessible from outside the cluster	Exist on a network that is accessible only within the cluster

Network Topology



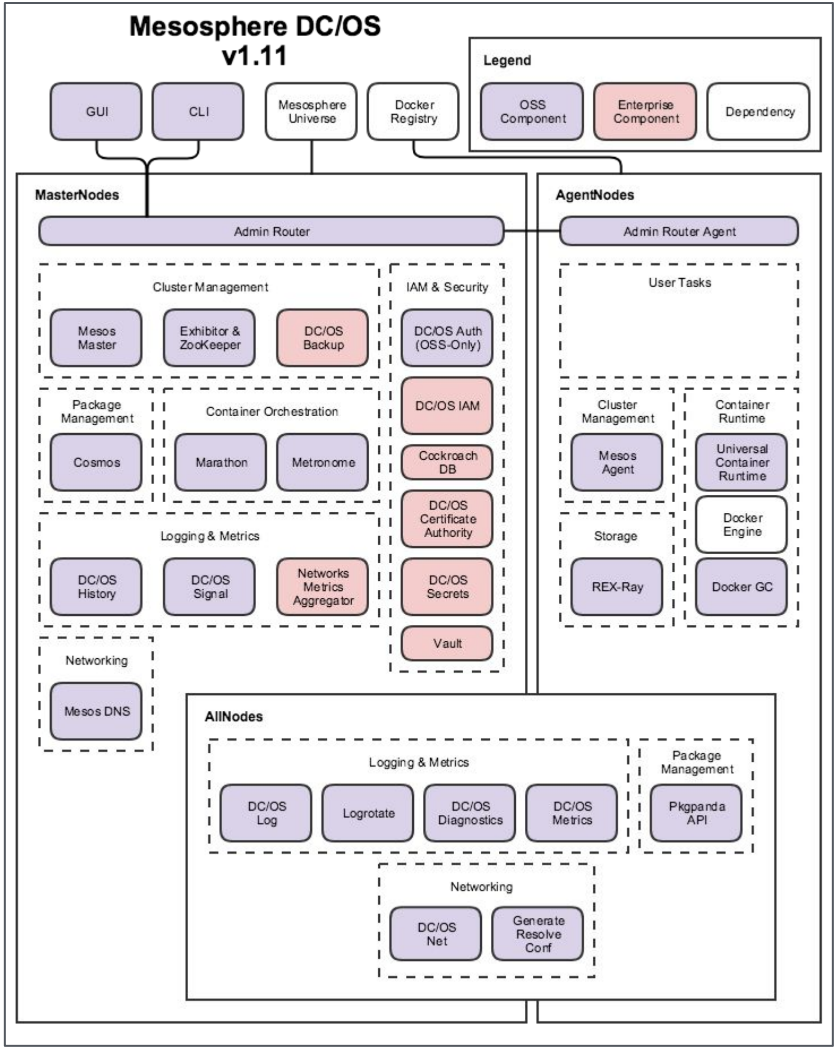
Bootstrap Node

- A single bootstrap node is required to install and later on upgrade the cluster
- The bootstrap node does not participate in the cluster it installs - only hosts the bits required to install/upgrade masters and agents
- Can be backed up and shut down after the installation is complete
- Minimum 2 core CPU and 16GB of RAM are required to service the installation process
- Must have open networking between bootstrap node and all potential cluster nodes

Software Requirements - All Nodes

- `tar`, `xz`, `unzip`, `curl`, `ipset`, and `docker` are required on all nodes
- SELinux must be either `disabled` or `placed` in permissive mode
    - Documentation
- DC/OS 1.11 supports Docker CE 17.05/17.06 and Docker EE 17.06
- See details for storage drivers and other Docker settings:
    - Documentation

Software Components



Mesosphere DC/OS v1.11

Legend

- OSS Component
- Enterprise Component
- Dependency

GUI

CLI

Mesosphere Universe

Docker Registry

MasterNodes

Admin Router

Cluster Management

- Mesos Master
- Exhibitor & ZooKeeper
- DC/OS Backup

IAM & Security

- DC/OS Auth (OSS-Only)
- DC/OS IAM
- Cockroach DB
- DC/OS Certificate Authority
- DC/OS Secrets
- Vault

Package Management

- Cosmos

Container Orchestration

- Marathon
- Metronome

Logging & Metrics

- DC/OS History
- DC/OS Signal
- Networks Metrics Aggregator

Networking

- Mesos DNS

AgentNodes

Admin Router Agent

User Tasks

Cluster Management

- Mesos Agent

Container Runtime

- Universal Container Runtime
- Docker Engine
- Docker GC

Storage

- REX-Ray

AllNodes

Logging & Metrics

- DC/OS Log
- Logrotate
- DC/OS Diagnostics
- DC/OS Metrics

Package Management

- Pkgpanda API

Networking

- DC/OS Net
- Generate Resolve Conf

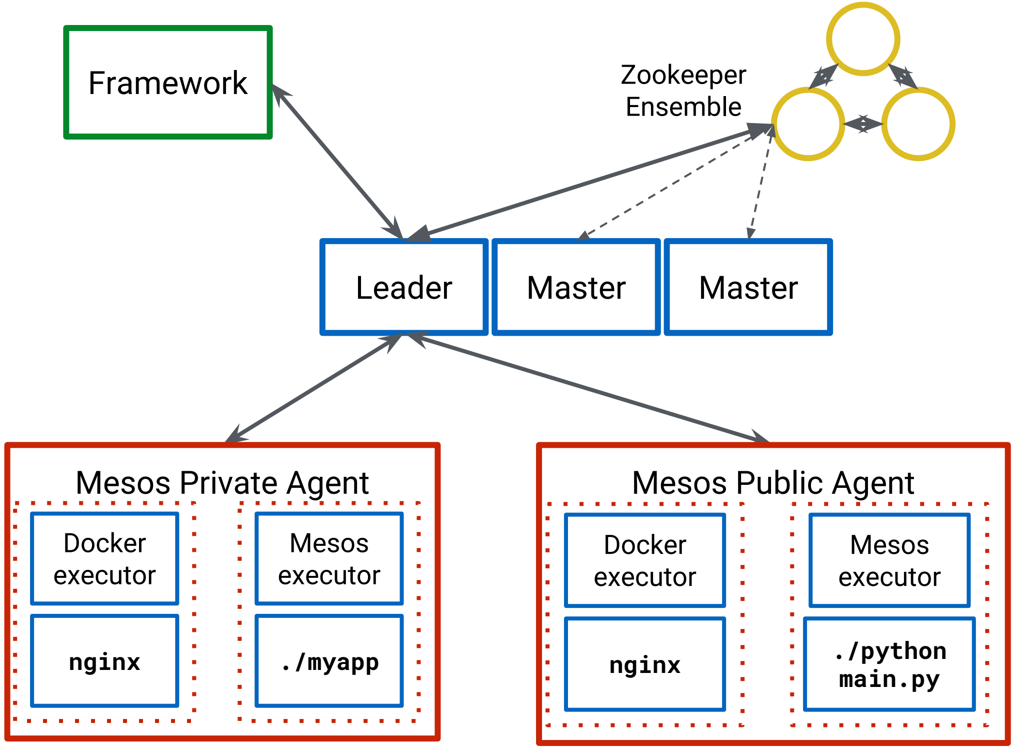
Lab 0

Build Your Cluster

DC/OS Cluster Upgrades

Upgrade Step 0: Do your Homework

1. Make sure cluster is healthy
2. Review release notes
3. Perform backup
   a. Zookeeper
   b. Replicated logs
   c. Application state



Backup and Restore

- Marathon state in ZK can be backed up using the DC/OS CLI
- Available to Enterprise customers
- Backups are managed by the `dcos-backup-master` component running on each master nodes
- Backups are stored on the local filesystem of the master(s)
- Must install the Enterprise CLI package to get the backup subcommand
  - `dcos package install --yes dcos-enterprise-cli`

Backup Procedure

1. Create backup: `dcos backup create --label=<backup_name>`
2. View progress and state: `dcos backup list`
3. View details of backup: `dcos backup show <backup_id>`
4. Delete old backups: `dcos backup delete <backup_id>`

Restore Procedure

1. Retrieve backup ID: `dcos backup list`
2. Begin restore: `dcosk backup restore <backup_id>`
3. View progress of restore: `dcos backup show <backup_id>`

Pre and Post Upgrade Diagnostics

Overview:

- Node and cluster health checks to determine state of services on the cluster
- Health checks are automatically executed when upgrading
- Included in the `dcos-diagnostics` bundle

Benefits:

- Helps prevent issues related to configuration drift
- Can check for host-level daemons that are not part of the base install of DC/OS

Post Upgrade Diagnostics

Overview:

- Diagnostics CLI is installed on each node in your cluster
- Absolute path: `/opt/mesosphere/bin/dcos-diagnostics`
- Execute `source dcos-shell` to set up environment variables that will put the executable in your path

Examples:

- `source dcos-shell`
- `dcos-diagnostics check node-poststart`
- `dcos-diagnostics check cluster`
- `dcos-diagnostics check node-poststart --list`
- `dcos diagnostics check cluster --list`

Custom Health Checks

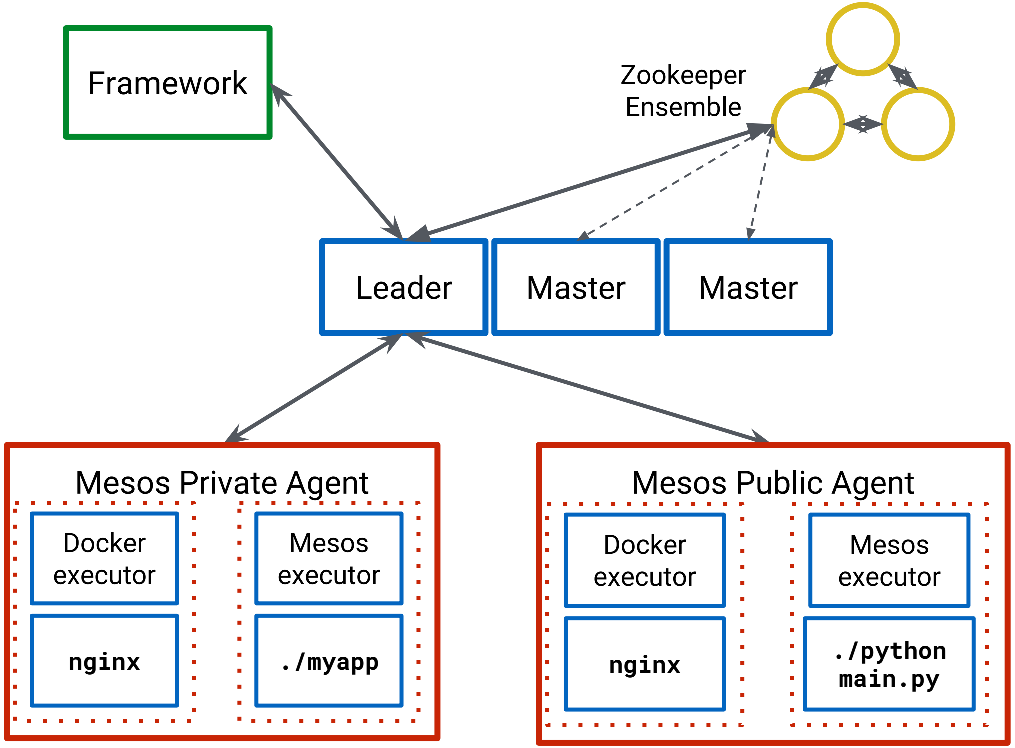
Overview:

- Configured in `config.yaml`
- Two types of checks: cluster and node
- Cluster checks report the health status of the entire DC/OS cluster
- Node checks report the status of individual nodes after installation
- Documentation

```
custom_checks: {
  ‘node_checks’: {
    ‘checks’: {
      ‘custom_node_check’: {
        ‘description’: ‘My custom check’,
        ‘cmd’: [‘echo’, ‘node check’],
        ‘timeout’: ‘2s’
      }
    }
  ‘poststart’: [‘custom-node-check’]
}
```

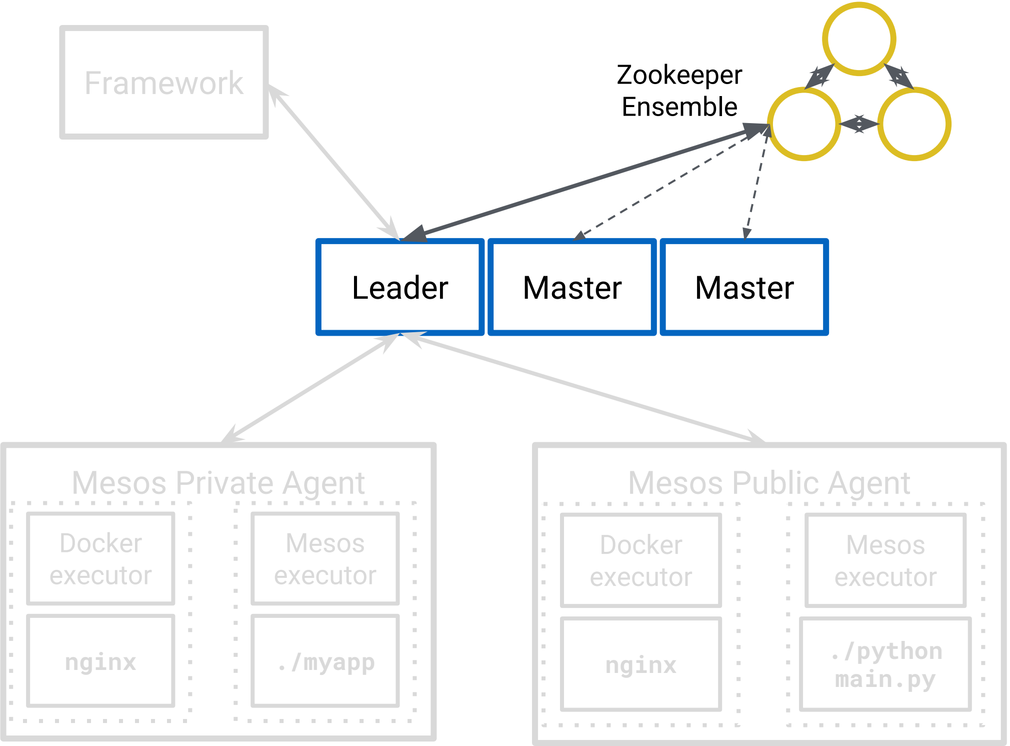
Upgrade Step 1: Generate Upgrade Artifacts

1. Download the DC/OS installer (`dcos_generate_config.ee.sh`) on to your bootstrap node
2. Create the `genconf/` directory and put a copy of your current cluster's `config.yaml`, `ip-detect`, and `license.txt` there
3. Generate upgrade artifacts
4. Serve upgrade artifacts on to the network over HTTP/HTTPS



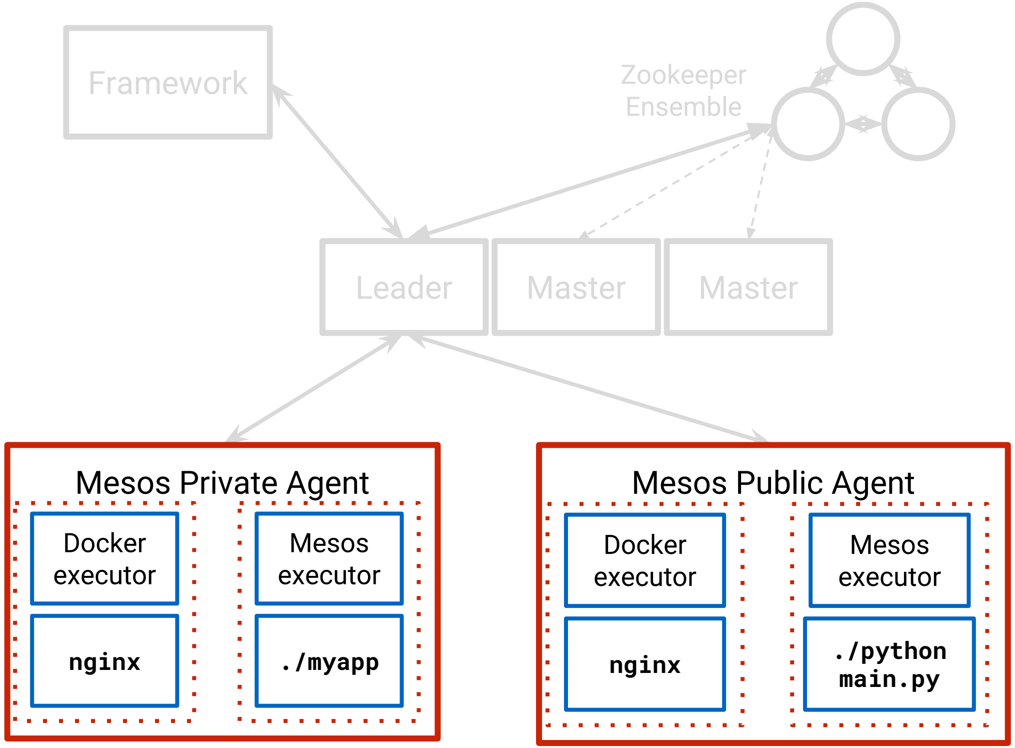
Upgrade Step 2: Master Nodes

1. Determine current set of standby Mesos master nodes
2. Upgrade each standby master node serially:
   a. Download upgrade script from bootstrap node
   b. Execute upgrade script
3. Upgrade leading Mesos master node once all other masters have been upgraded



Upgrade Step 3: Agent Nodes

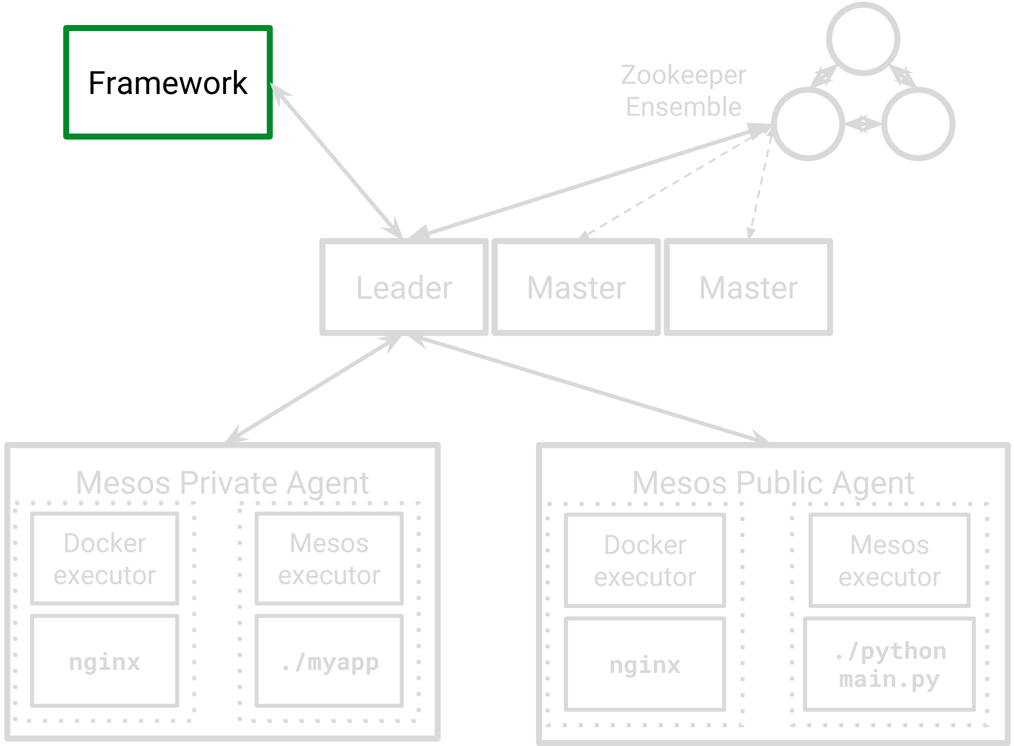
1. Upgrade agents:
   a. Download upgrade script from bootstrap node
   b. Execute upgrade script



Multiple agents can be upgraded in parallel, however it is wise to upgrade in smaller batches to ensure application availability is not impacted should an error be encountered

Upgrade Step 4: Catalog Packages

- Similar to how new versions of DC/OS are released, packages from the Catalog have their own release cadence
- Some packages have updates that can be rolled out when a new version of DC/OS is put in play
- Consult the service’s documentation for more details



Labs

1. Upgrading DC/OS
2. Backup and Restore

Log Management

DC/OS Logs

- DC/OS software components use `journald` for managing log output
- Two main types of logs in the cluster
  - DC/OS component logs: Logs for all DC/OS software components running on masters and agents (e.g. `dcos-mesos-master`, `dcos-mesos-agent`, `dcos-marathon`, etc.)
  - Userspace logs: `stdout` and `stderr` for all containers running on agents
- Log rotation is configured automatically during install

Accessing Logs: DC/OS CLI

- DC/OS CLI is capable of pulling logs over HTTP/HTTPS
- Node level logs:
  - `dcos node log --leader --lines 60`
  - `dcos node log --leader --follow`
  - `dcos node log --master-proxy --mesos-id <agent_id> --follow`
- Framework level logs:
  - `dcos service log cassandra`
  - `dcos service log --follow hdfs`

Accessing Logs: DC/OS CLI

- To retrieve stdout or stderr for a particular container, retrieve list of containers:
  - `dcos task`
- Copy “ID” column for container you are interested in
- Retrieve `stdout` from given task:
  - `dcos task log <id>`
- Retrieve `stderr` from given task:
  - `dcos task log <id> stderr`

Accessing Logs: Logging API

- The DC/OS Logging API is a REST API that runs on each node in the cluster that allows for log retrieval over HTTP/HTTPS
- Exposes node, component, and container logs
- Supports multiple output formats:
  - `text/plain`
  - `application/json`
  - `text/event-stream`
- Requires valid authorization token

Accessing Logs: External Log Aggregation

- Log aggregation is a must for any long-lived DC/OS cluster
- Can leverage pre-existing log aggregation systems such as Splunk and the ELK stack
- High Level Steps:
  - Install appropriate forwarder (Universal forwarder, logstash, etc.) on all nodes in the cluster
  - Create a `logstash.conf` file for ELK
  - Create a shell script which retrieves all DC/OS component logs with `journalctl`
    - ELK: Start the logstash daemon
    - Splunk: Add the shell script as an input source to the forwarder
  - On agent nodes, create an input source which collects all stdout and stderr files found below `/var/lib/mesos`

Labs

3. DC/OS Logging
4. DC/OS Logging API
5. Log Aggregation with Splunk

Lab & lunch until
1:20 PM CST
8:20 PM CET

MESOSPHERE