

# MESOSPHERE

## Replacing DC/OS Master Nodes

*Master node replacement or expansion in DC/OS*

*August 28, 2018*

*Arthur Johnson,*

*Solutions Architect, Mesosphere Inc.*

[ajohnson@mesosphere.io](mailto:ajohnson@mesosphere.io)

<b>Overview and Motivation</b>	<b>3</b>
<b>High-Level Process</b>	<b>4</b>
<b>Determining Master Node Replacement</b>	<b>5</b>
Determining ZooKeeper IDs and Correlation to IPs	5
A Working Example	6
<b>Node Replacement - Starting State</b>	<b>8</b>
<b>Node Replacement from a Catastrophic Failure</b>	<b>13</b>
<b>Expanding a 3-node Master Cluster to 5-nodes</b>	<b>15</b>
<b>Document Versioning</b>	<b>18</b>

# Overview and Motivation

This document describes the details of either replacing Master Nodes or increasing the count of Masters in a currently running DC/OS 1.9.10 cluster. Generally speaking, Masters are the most critical architecture component in a DC/OS cluster and must be carefully planned for and implemented. If the expectation is to have 5 Masters in a cluster, it is most efficient and risk adverse to implement the correct number of Masters from the beginning installation of the cluster.

DC/OS clusters are meant to be long-running entities which should prevail over the tests of both time and changes in the data center environment. However, there are cases where hardware fails, requirements change, and rebalancing of resources must be done.

In the course of this document we will perform two configuration changes to meet two distinct objectives, and a third (catastrophic replacement induced during an error in our test environment):

Configuration Change	Objective	Requirements
Replace existing Master(s)	Remove a faulted Master and install a permanent replacement	Gathering current cluster state; ascertain ZK state and configuration; provision replacement node(s); install DC/OS; upgrade DC/OS on the remaining nodes in the cluster
Add more Master(s) to a currently running configuration	Take a 3-node Master cluster to 5-nodes	Gather current cluster state; ascertain ZK state and configuration; provision additional nodes; install DC/OS masters; upgrade remaining nodes in the cluster

The two procedures are very similar-- however, care must be taken to make sure that the ZK IDs of the current Master nodes in the cluster are not duplicated or used in an 'out-of-lexical-order'.

# High-Level Process

At a high level, this process follows the following steps:

1. Determine the correct order to replace the Masters. If Masters are replaced in the wrong order, you will experience outages during the migration process (you may still experience outages, but they should be minimal).
2. Then, for each master replacement:
  - a. Update the `genconf/config.yaml` with the new master (removing the old master)
  - b. Generate a node upgrade script using `dcos_generate_config.ee.sh --generate-node-upgrade-script <version>`, where `<version>` is the current version of DC/OS. This will automatically re-create the regular `dcos_install.sh` script.
  - c. Install DC/OS on the new master, using `dcos_install.sh master`
  - d. Stop DC/OS on the old master (the one that is being replaced), using `sudo systemctl stop -- $(systemctl show -p Wants dcos.target | cut -d= -f2)`
  - e. For each existing master, complete the following, one at a time:
    - i. Use the newly-generated upgrade script to upgrade the master
    - ii. Wait for zookeeper to settle and re-replicate (zxid should match)
    - iii. Wait for Mesos and Overlay to the replica log to re-replicate
    - iv. Validate that all checks work properly
    - v. Optionally, wait an additional 15 minutes
    - vi. Continue to the next master
  - f. Validate that all certificates have been properly propagated to the new master
  - g. Validate that all checks complete successfully on the new master
  - h. Update all of the private and public slaves to point to the new masters
    - i. Validate that `docker ps` returns successfully
    - ii. Run the node upgrade script
    - iii. Verify that the node returns from its update as healthy
  - i. Wait an hour to soak the changes (Optional, depending on state of activity in the cluster)
3. Once all of the masters have been updated, ensure that all external DNS changes have been made

# Determining Master Node Replacement

The first step in replacing masters is determining the correct order in which to perform the replacement -- if Masters are replaced in an incorrect order there is risk that Exhibitor will experience outages due to different ZK instances that will have non-matching server-id to server mapping.

Here's the key objective for this activity is: we want to replace masters one at a time, following three rules:

1. The ZK server id order should match that generated by the installation script (dcos\_node\_upgrade script)
2. The ZK server id for a given IP server should never change
3. No two servers should have non-matching ZK server ids for a given server

Every time you run the dcos\_node\_upgrade script, a new order is generated - we don't want this order to ever change, aside from the one node we're changing at that time.

If the three rules are not respected, and you replace Masters out of the above order, the cluster will eventually recover... However, you will have multiple outages and service interruption until the cluster settles. This is not a favorable situation for the cluster to be in...

Exhibitor (dcos-exhibitor) uses python's 'sort' function to determine the order it will use for ZK; this does a raw ASCII sort, disregarding the IP address format.

*(The code used to generate this list is viewable in the calculate\_exhibitor\_static\_ensemble command in the gen/calc.py file in the Github repository dcos/dcos: <https://github.com/dcos/dcos/blob/master/gen/calc.py#L465>)*

## Determining ZooKeeper IDs and Correlation to IPs

1. Determine the correct ZooKeeper order for the current list of IPs
2. Determine the correct ZooKeeper order for the new list of IPs
3. Choose a Master to replace that is in the same position in both the new list and old list, such that replacing it will not disrupt the order (i.e., after replacing it and generating a new list, the list will still be in order). There may be multiple options here - any of them are fine, as long as they don't violate this rule
4. Repeat step "3" until you come up with a valid order of masters to replace

## A Working Example

Assume we're replacing IPs:: 10.10.0.236, 100.10.0.66, 10.10.0.158, 100.10.0.158, 10.10.0.149 with these IPs: 100.10.0.85, 100.10.0.147, 10.10.0.230, 10.10.0.184, 10.10.0.233

Then you're starting with these ZK server ids (as determined by python sort; note that '100.10.0.66' comes after '100.10.0.158' because '6' is after '1', and that '100.10.0.158' comes after '10.10.0.236' because '.' comes before '0' lexically, i.e., following ASCII character codes):

```
1: 10.10.0.149
2: 10.10.0.158
3: 10.10.0.236
4: 100.10.0.158
5: 100.10.0.66
```

and moving to:

```
1: 10.10.0.184
2: 10.10.0.230
3: 10.10.0.233
4: 100.10.0.147
5: 100.10.0.85
```

We can't yet replace Master 1, because that would put 1 and 2 out of lexical order:

```
1: 10.10.0.184
2: 10.10.0.158
3: 10.10.0.236
4: 100.10.0.158
5: 100.10.0.66
```

We must first replace Master 2, because it doesn't break lexical order (10.10.0.230 is after 10.10.0.149 and before 10.10.0.236):

```
1: 10.10.0.149
2: 10.10.0.230 (was 10.10.0.158)
3: 10.10.0.236
4: 100.10.0.158
5: 100.10.0.66
```

We can now replace Master 1, because it will no longer break lexical order (10.10.0.184 is before the new 10.10.0.230):

```
1: 10.10.0.184
2: 10.10.0.230
3: 10.10.0.236
4: 100.10.0.158
```

5: 100.10.0.66

And so forth. After this, we can replace masters 3, 4, and 5, because this order will not change any ZK server IDs.

# Node Replacement - Starting State

This procedure was done in the following environment:

- CoreOS 1632.3.0
- DC/OS 1.9.10
- 5 Masters

Masters must be replaced one at a time; and, between each master replacement, all nodes must be upgraded (configuration change) to point to the newly added, or removed, Masters.

During and throughout this process, care must be taken to identify any services either within or external to the cluster that may have used hardcoded Master IPs. In the event that IP addresses change, services will be negatively affected.

If we are just replacing a node, or adding two new ones, the `dcos_node_upgrade` script and `dcos_install` script will take the burden of calculating ZK node/IP address and ensure that the order is maintained.

Check the health of the cluster, and make sure that the cluster is stable and all master nodes are in a healthy serving state: Verify the state of Exhibitor:

```
a. curl -fsSL  
    http://localhost:8181/exhibitor/v1/cluster/status |jq .
```



```
core@ip-172-31-31-63 ~ $ curl -fsSL http://localhost:8181/exhibitor/v1/cluster/status |jq .
[
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.16.60",
    "isLeader": true
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.18.152",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.29.145",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.29.82",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.31.63",
    "isLeader": false
  }
]
core@ip-172-31-31-63 ~ $
```

If there are any issues, investigate the health of Exhibitor  
`sudo journalctl -flu dcos-exhibitor`

It is preferred to replace a master node which is not currently the elected leader to minimize the number of elections that the cluster will experience.

We are going to replace up-to two Masters in this procedure.

The order of the masters can be found in the cluster configuration, on the masters, at the following location:

`/opt/mesosphere/etc/exhibitor`

The contents of exhibitor will show the servers in their current order:

“Exhibitor”

```
core@ip-172-31-31-63 /opt/mesosphere/etc $ cat exhibitor
```

```
EXHIBITOR_BACKEND=STATIC
```

```
EXHIBITOR_STATICENSEMBLE=1:172.31.16.60,2:172.31.18.152,3:172.31.29.145,4:172.31.29.82,5:172.31.31.63
```

In this example, leader with the IP of: 172.31.16.60 is ID #1, and the current leader.

We are going to replace servers #5, 172.31.31.63 and #4, 172.31.29.82.  
We will start with replacing #5.

Edit the cluster '`config.yaml`' and location the IP of the server with ID #5. You'll remove this IP address from the config and replace it with the server being added to the cluster.

```
master list:
- 172.31.31.63
- 172.31.18.52
- 172.31.29.145
- 172.31.29.82
- 172.31.16.60
resolvers:
- 169.254.169.253
- 8.8.8.8
```

We will do one server at a time, ensuring that order persists and that the removal and introduction of new masters does not disturb the cluster (or cause an outage).

After updating the "`genconf/config.yaml`" script with the IP address of the replacement server, generate a "node upgrade script":

```
sudo bash dcos_generate_config.ee.sh --generate-node-upgrade-script
1.9.10
```

The node upgrade script will be used to 'update/upgrade' the nodes throughout the cluster.

```
core@ip-172-31-16-198 ~ $ ls
dcos-genconf.aa8d31e44e1d75400e-d730f3325d4b75596b.tar  eth0.masters.addrs  key.pem
dcos_generate_config.ee.sh                               genconf             master.list
core@ip-172-31-16-198 ~ $ vi genconf/config.yaml
core@ip-172-31-16-198 ~ $ sudo bash dcos_generate_config.ee.sh --generate-node-upgrade-script 1.9.10
Generating configuration files...
Generating configuration files...
Package filename: packages/dcos-config/dcos-config--setup_624b614c529880fa633e2d48906e12fdddc7601f.tar.xz
Package filename: packages/dcos-metadata/dcos-metadata--setup_624b614c529880fa633e2d48906e12fdddc7601f.tar.xz
Generating Bash configuration files for DC/OS
Node upgrade script URL: http://172.31.16.198:80/upgrade/300b3a91f27c4ceb9e9aef43e04d0fc5/dcos_node_upgrade.sh
core@ip-172-31-16-198 ~ $
```

Install DC/OS on the new master, as you normally would with:  
`sudo bash dcos_install.sh master`

Stop the master services running on the old master, being replaced:

```
sudo systemctl stop -- $(systemctl show -p Wants dcos.target | cut -d= -f2)
```

After the old master has been removed, and the new master installed. Upgrade the remaining masters one at a time.

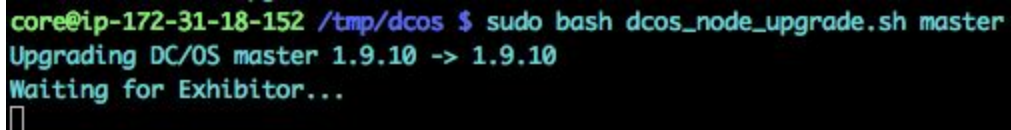
Download the node upgrade script:

```
curl -O  
http://172.31.16.198:80/upgrade/300b3a91f27c4ceb9e9aef43e04d0fc5/dcos_node_upgrade.sh
```

Apply the upgrade:

```
Sudo bash dcos_node_upgrade.sh master
```

You will notice there is a moment or two for Exhibitor to settle:

A terminal window with a black background and green text. The prompt is 'core@ip-172-31-18-152 /tmp/dcos \$'. The command entered is 'sudo bash dcos\_node\_upgrade.sh master'. The output shows 'Upgrading DC/OS master 1.9.10 -> 1.9.10' followed by 'Waiting for Exhibitor...' and a cursor on a new line.

```
core@ip-172-31-18-152 /tmp/dcos $ sudo bash dcos_node_upgrade.sh master  
Upgrading DC/OS master 1.9.10 -> 1.9.10  
Waiting for Exhibitor...  
█
```

Check exhibitor state:

```
core@ip-172-31-18-152 /tmp/dcos $ curl -fsSL http://localhost:8181/exhibitor/v1/cluster/status |jq .
```

```
[
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.16.60",
    "isLeader": true
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.18.152",
    "isLeader": false
  },
  {
    "code": 2,
    "description": "not serving",
    "hostname": "172.31.22.240",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.29.145",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.29.82",
    "isLeader": false
  }
]
```

**New master**

Move on to the remain 3 masters...

Doing the leading master last, as to not have unnecessary elections!

You can watch the election/status by putting the `curl -fsSL` to exhibitor in a loop. If there isn't much going on in the cluster, the election will happen very quickly and the cluster will stabilize.

Once masters are done, give the cluster ~15-20 minutes of time to stabilize.

Next, move onto the agents, throughout the cluster and run the same `node_upgrade_script` on them passing the correct argument for their personality (private or public slave).

# Node Replacement from a Catastrophic Failure

Handling an outage, where a master server has been removed or experiences a catastrophic failure causing it to no longer be part of the cluster:

```
curl -fsSL http://localhost:8181/exhibitor/v1/cluster/status |jq .
[
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.16.60",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.18.152",
    "isLeader": false
  },
  {
    "code": 1,
    "description": "down",
    "hostname": "172.31.22.240",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.29.145",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.29.82",
    "isLeader": true
  }
]
```

This node is “**down**”, no longer serving, and no longer available.

This node can be replaced via the same method of swapping out a running master.

- Edit the `config.yaml`, removing the “down” server, replacing with a fresh server
- Generate the upgrade script
- Install the master via `sudo bash dcos_install.sh master`
- Upgrade the surviving masters via `sudo bash dcos_node_upgrade.sh master`
- Wait for the cluster to stabilize

- Upgrade the agent nodes via `sudo bash dcos_node_upgrade.sh slave (slave_public)`

# Expanding a 3-node Master Cluster to 5-nodes

Start with 3 masters, in a cluster, normal, healthy state:

```
core@ip-172-31-27-57 /tmp/dcos $ curl -fsSL http://localhost:8181/exhibitor/v1/cluster/status |jq .
[
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.27.182",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.27.57",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.31.133",
    "isLeader": true
  }
]
```

Provision two additional hosts to become master #4 and master #5.

Add the two new nodes to config.yaml - vi genconf/config.yaml

- New masters:

```
master_list:
- 172.31.27.57
- 172.31.31.133
- 172.31.27.182
- 172.31.30.129
- 172.31.30.220
```

Generate a new dcos\_node\_install and upgrade script:

```
sudo bash dcos_generate_config.ee.sh --generate-node-upgrade-script
1.9.10
```

On the new masters, install as normal:

```
sudo bash dcos_install.sh master
```

- Wait until they are installed, and showing in ZK state before performing the upgrade on the existing masters

On the three starting/existing masters, apply the new configuration via an upgrade:

**Sudo bash dcos\_node\_upgrade.sh master**

- While these upgrades are in progress, zk state will show no leader, latent state, not serving. But as the cluster elections take place, and Zookeeper settles, the cluster will resume a normal serving state

```
core@ip-172-31-27-182 /tmp/dcos $ curl -fsSL http://localhost:8181/exhibitor/v1/cluster/status |jq .
[
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.27.182",
    "isLeader": false
  },
  {
    "code": 1,
    "description": "down",
    "hostname": "172.31.27.57",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.30.129",
    "isLeader": true
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.30.220",
    "isLeader": false
  },
  {
    "code": 1,
    "description": "down",
    "hostname": "172.31.31.133",
    "isLeader": false
  }
]
```

Return to a normal, healthy state:



```
[
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.27.182",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.27.57",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.30.129",
    "isLeader": true
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.30.220",
    "isLeader": false
  },
  {
    "code": 3,
    "description": "serving",
    "hostname": "172.31.31.133",
    "isLeader": false
  }
]
```

Proceed to upgrade the agents so that they are made aware of the expanded master set

- Sudo bash dcos\_node\_upgrade.sh slave (slave\_public)

# Document Versioning

Document revisions, version, date, details:

Version	Date	Details
1	August 28, 2018	Initial draft - ajohnson@mesosphere.com;
2	November 9, 2018	Minor updates