Table of Contents

[Table of Contents 1](#_Toc496168948)

[1 Install dcos commands 1](#_Toc496168949)

[2 Config User 1](#_Toc496168950)

[3 Install Marathon-lb 2](#_Toc496168951)

[4 Install Private Registry 2](#_Toc496168952)

[4.1 Create TLS Certificate 2](#_Toc496168953)

[4.2 Setup Web Server Serving Certificate 3](#_Toc496168954)

[4.3 Create Certs on Nodes 3](#_Toc496168955)

[4.4 Create config.yml 3](#_Toc496168956)

[4.5 Deploy Registry 4](#_Toc496168957)

[4.6 Update marathon-lb 4](#_Toc496168958)

[4.7 Test registry 4](#_Toc496168959)

[5 Kafka 4](#_Toc496168960)

[5.1 Setup confluent-kafka 4](#_Toc496168961)

[5.2 Setup schema-registry 5](#_Toc496168962)

[5.3 Setup rest-proxy 5](#_Toc496168963)

[5.4 Setup control-center 5](#_Toc496168964)

# Install dcos commands

Copy dcos to 1st master node

> sudo su -

Move dcos to /usr/local/bin/

> dcos config set core.dcos\_url <https://10.165.105.151>

> dcos config set core.ssl\_verify false

> dcos package install dcos-enterprise-cli

# Config User

We have 3 users to handle: (admin-br, readonly, bootstrapuser)

1. Add new user “**admin-gr**” to superusers group 同样命名

2. Change password of **bootstrapuser** 这个要保留

Logout

Login with admin-gr

3. Create readonly/readonly, give it permission (copy from AL ship)

Or try following manually:

**!!! ALL secrets, password, service account should NOT be added in this document !!!**

**readonly** User:

readonly:readonly

dcos:adminrouter:ops:mesos | full (Task Tab)

dcos:adminrouter:ops:slave | full (Task Detail)

dcos:adminrouter:package | full (Universe Tab)

dcos:adminrouter:service:marathon | full (Service Tab)

dcos:adminrouter:ops:networking | full (Network Tab)

dcos:adminrouter:service:metronome | full (Jobs Tab)

dcos:adminrouter:ops:system-health | full (System Tab)

dcos:service:marathon:marathon:services:/ | read (services)

dcos:service:metronome:metronome:jobs | read

**Or use the Advanced mode to do bulk editing: ( same effect as above manual way)**

dcos:adminrouter:ops:mesos full

dcos:adminrouter:ops:networking full

dcos:adminrouter:ops:slave full

dcos:adminrouter:ops:system-health full

dcos:adminrouter:package full

dcos:adminrouter:service:marathon full

dcos:adminrouter:service:metronome full

dcos:service:marathon:marathon:services:/ read

dcos:service:marathon:marathon:services:/confluent read

dcos:service:marathon:marathon:services:/confluent-kafka full

dcos:service:metronome:metronome:jobs read

# Install Marathon-lb

Copy **create-marathon-lb-accounts.sh**, **marathon-lb.json**, and **install-marathon-lb.sh** to /home/mesossu/rcclapps/marathon-lb folder (create the folder if not present) of master node 1 (10.165.105.151).

Run the following command that will create the keys and secret for installation

**sudo su –**

**./create-marathon-lb-accounts.sh （在新版本mesos, 最哈重新装 CLI， Security包）**

Run the following command that will install marathon lb

**./install-marathon-lb.sh 别忘了改json**

To test the installation, browse <http://10.165.105.158:9090/haproxy?stats> that will show ha proxy page

# Install Private Registry

DNS: registry.grandeur.sh.rccl.com (.157)

这一步不着急 Steps: (similar to step 4.5)

1. copy docker registry JSON from AL

2. modify it (ip, dns, 等等)

3. install it by going to dashboard, Services->service, click ‘+’, choose ‘JSON Configuratioin’ to start.

> ping registry.grandeur.sh.rccl.com # from any node in current cluster see if it’s resolved

TODO: no F5 balancer ready, we cannot use above NS.

## Create TLS Certificate

On 1st master node:  
(in su) sudo su -  
**mkdir /home/mesossu/rcclapps/registry** && **cd /home/mesossu/rcclapps/registry**

**vi /home/mesossu/rcclapps/registry/gen-cert.sh**

做一个文件，可能可以在boot node上做，没有

#!/bin/bash

#repo\_url length needs to be 64 or less

REPO\_URL=registry.mariner.sh.rccl.com

echo "Generating key and crt"

openssl req -newkey rsa:4096 -nodes -sha256 \

-keyout domain.key -x509 -days 3650 \

-out domain.crt \

-subj "/C=US/ST=Florida/L=Miami/O=IT/CN=${REPO\_URL}"

echo "Generating pem"

cat domain.crt domain.key | tee registry.pem

**chmod 700 /home/mesossu/rcclapps/registry/gen-cert.sh**

**/home/mesossu/rcclapps/registry/gen-cert.sh**

## Setup Web Server Serving Certificate

On **Bootstrap** server: (this can be changed to use other node or LoadBalancer for high availability)

sudo su -

mkdir /webserver

cd /webserver

place domain.crt , domain.key and the registry.pem files

python -m SimpleHTTPServer 8404 &> /dev/null &

## Create Certs on Nodes

On all private and public agents:

**sudo su -**

**vi copycrt.sh**

**做在jump 上即可，**

#!/bin/bash

DOMAIN\_CRT=domain.crt

REPO\_URL=registry.mariner.sh.rccl.com

REPO\_PORT=10104

rm -f domain.crt\* registry.pem\*

wget http://10.132.105.150:8404/domain.crt

wget http://10.132.105.150:8404/registry.pem

mkdir -p /etc/docker/certs.d/${REPO\_URL}:${REPO\_PORT}

cp ${DOMAIN\_CRT} /etc/docker/certs.d/${REPO\_URL}:${REPO\_PORT}/ca.crt

# RHEL

cp ${DOMAIN\_CRT} /etc/pki/ca-trust/source/anchors/${REPO\_URL}.crt

echo "Updating CA trust"

update-ca-trust

# This is for DCOS version 1.8 and lower only

CACERT=/opt/mesosphere/active/python-requests/lib/python3.5/site-packages/requests

echo "Hack for 1.8 cacerts"

cp ${CACERT}/{cacert.pem,cacert.pem.bup}

cat registry.pem >> ${CACERT}/cacert.pem

systemctl restart docker

**回头一个一个copy上去**

**Or following mesos 1.11 (not ready, 试一试)**

#!/bin/bash

DOMAIN\_CRT=domain.crt

REPO\_URL=registry.mariner.sh.rccl.com

REPO\_PORT=10104

rm -f domain.crt\* registry.pem\*

wget http://10.196.105.190:8404/domain.crt

wget http://10.196.105.190:8404/registry.pem

mkdir -p /etc/docker/certs.d/${REPO\_URL}:${REPO\_PORT}

cp ${DOMAIN\_CRT} /etc/docker/certs.d/${REPO\_URL}:${REPO\_PORT}/ca.crt

# RHEL

cp ${DOMAIN\_CRT} /etc/pki/ca-trust/source/anchors/${REPO\_URL}.crt

echo "Updating CA trust"

update-ca-trust

# This is for DCOS version 1.8 and lower only

# CACERT=/opt/mesosphere/active/python-requests/lib/python3.5/site-packages/requests

# Following is uncertain yet:

# for DCOS 1.11

cp /opt/mesosphere/lib.old/python3.5/site-packages/requests/cacert.pem \

/opt/mesosphere/active/python-requests/lib/python3.6/site-packages/requests

CACERT=/opt/mesosphere/active/python-requests/lib/python3.6/site-packages/requests

echo "Hack for 1.8 cacerts"

cp ${CACERT}/{cacert.pem,cacert.pem.bup}

cat registry.pem >> ${CACERT}/cacert.pem

systemctl restart docker

chmod 700 copycrt.sh

./copycrt.sh

**偷懒做法**:

1) create above copycrt.sh on jumpbox:/tmp/copycrt.sh (modify the ipaddr)

2)

scp -q copycrt.sh 10.157.105.151:/tmp && \

scp -q copycrt.sh 10.157.105.152:/tmp

scp -q copycrt.sh 10.157.105.153:/tmp && \

scp -q copycrt.sh 10.157.105.154:/tmp && \

scp -q copycrt.sh 10.157.105.155:/tmp

scp -q copycrt.sh 10.157.105.156:/tmp && \

scp -q copycrt.sh 10.157.105.157:/tmp && \

scp -q copycrt.sh 10.157.105.158:/tmp && \

scp -q copycrt.sh 10.157.105.159:/tmp

**偷懒做法**:

ssh -t 10.157.105.151 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.152 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.153 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.154 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.155 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.156 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.157 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.158 "sudo /tmp/copycrt.sh"

ssh -t 10.157.105.159 "sudo /tmp/copycrt.sh"

## Create config.yml

Login 4th private node alcmesagt934 (.157)

sudo su -

mkdir -p /var/lib/docker/registry

cd /var/lib/docker/registry

vi config.yml # copied from AL, modify the “ship-al” to “ship-gr” and ip , and ip regex.

## Deploy Registry

Initial deploy through UI

Name: shared-services/registry

PIN to the Registry server

Environment Variable: "DOCKER\_OPTS": "--insecure-registry http://10.154.105.157:5000"

HOST\_VOLUME\_REGISTRY: /var/lib/docker/registry/shared-services/registry

Optional URLs: http://10.154.105.150:8404/domain.crt, http://10.154.105.150:8404/domain.key

Optional Constraints: hostname:CLUSTER:10.154.105.157

"HAPROXY\_GROUP": "external",

"HAPROXY\_0\_SSL\_CERT": "/mnt/mesos/sandbox/registry.pem",

"HAPROXY\_0\_BACKEND\_REDIRECT\_HTTP\_TO\_HTTPS": "false",

"HAPROXY\_0\_VHOST": "10.154.105.158,10.154.105.159,registry.allure.sh.rccl.com"

Update service ports to 10104 (2 places)

## Update marathon-lb

Update mlb-al-ext

Optional | URLs

<http://10.154.105.150:8404/registry.pem>

Restart mlb-al-ext

Add “fetch” attribute

"fetch": [

{

"uri": "http://10.132.105.150:8404/registry.pem",

"extract": true,

"executable": false,

"cache": false

}

],

## Test registry

<https://registry.grandeur.sh.rccl.com:10104/v2/_catalog>

<https://registry.brilliance.sh.rccl.com:10104/v2/_catalog>

<https://registry.mariner.sh.rccl.com:10104/v2/_catalog>

# Kafka

## Setup confluent-kafka

Copy **create-confluent-kafka-accounts.sh**, **install-confluent-kafka.sh**, and **confluent-kafka.json** to /home/mesossu/rcclapps/confluent-kafka folder on master node 1

Run **create-confluent-kafka-accounts.sh** that will create service accounts for confluent kafka

Run **./install-confluent-kafka.sh** that will install confluent kafka

Using Dashbaord, add the following json properties

"KAFKA\_OVERRIDE\_CONFLUENT\_METRICS\_ENABLE": "true",

"KAFKA\_OVERRIDE\_KAFKA\_METRIC\_REPORTERS": "io.confluent.metrics.reporter.ConfluentMetricsReporter",

"KAFKA\_OVERRIDE\_CONFLUENT\_METRICS\_REPORTER\_MAX\_REQUEST\_SIZE": "10485760",

Using Dashbaord, update the following json properties:

"KAFKA\_OVERRIDE\_DEFAULT\_REPLICATION\_FACTOR": "3",

"KAFKA\_OVERRIDE\_COMPRESSION\_TYPE": "snappy", //old value is ‘producer’

"KAFKA\_OVERRIDE\_NUM\_PARTITIONS": "5",

++ set “PHASE\_STRATEGY” to ‘STAGE’ --- this one was ‘INSTALL’ , chaging it to ‘STAGE’ caused ‘Deploying’ problem.

**Using kafka-client docker, run the following commands**:

**# Under Kafka-client**

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-configs --replication-factor 3 --partitions 1 --config cleanup.policy=compact'

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-offsets --replication-factor 3 --partitions 50 --config cleanup.policy=compact'

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-status --replication-factor 3 --partitions 10 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-cdc-configs --replication-factor 3 --partitions 1 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-cdc-offsets --replication-factor 3 --partitions 50 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-cdc-status --replication-factor 3 --partitions 10 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-ga-configs --replication-factor 3 --partitions 1 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-ga-offsets --replication-factor 3 --partitions 50 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-ga-status --replication-factor 3 --partitions 10 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-ncp-configs --replication-factor 3 --partitions 1 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-ncp-offsets --replication-factor 3 --partitions 50 --config cleanup.policy=compact'

docker run -it dev2.registry.rccl.com/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-ncp-status --replication-factor 3 --partitions 10 --config cleanup.policy=compact'

**# Under dcos CLI**

dcos confluent-kafka topic create \_schemas --partitions=1 --replication=3

**# or**

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c './kafka-topics.sh --create --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic '\_schemas' --replication-factor 3 --partitions 1'

**# Under Kafka-client**

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c './kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic \_schemas --alter --config cleanup.policy=compact'

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c './kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic \_schemas --alter --config min.insync.replicas=2'

**# Validate under Kafka-client**

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka -describe --topic dcos-connect-configs'

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka -describe --topic dcos-connect-offsets'

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka -describe --topic dcos-connect-status'

docker run -it registry.brilliance.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka -describe --topic \_schemas'

**# my other testing**

docker run -it registry.serenade.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-topics.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka -describe --topic dcos-connect-offsets'

docker run -it registry.serenade.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-console-consumer.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic dcos-connect-offsets --from-beginning'

docker run -it registry.serenade.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-console-**consumer**.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic **commerce\_product\_booking\_deltas\_v1** --from-beginning'

docker run -it registry.serenade.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-console-producer.sh --broker-list 10.130.105.155:9092 --topic **commerce\_product\_booking\_deltas\_v1** '

docker run -it registry.allure.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-console-**consumer**.sh --zookeeper master.mesos:2181/dcos-service-confluent-kafka --topic **commerce\_product\_booking\_deltas\_v1** --from-beginning'

docker run -it registry.allure.sh.rccl.com:10104/kafka-client /bin/bash -c 'kafka-console-**producer**.sh --broker-list 10.154.105.155:9092 --topic **commerce\_product\_booking\_deltas\_v1** '

kafka-console-producer.sh --broker-list 10.154.105.154:9092,10.154.105.155:9092,10.154.105.156:9092 --topic test11

kafka-console-consumer.sh --zookeeper 10.154.105.153:2181/dcos-service-confluent-kafka --topic test11 --from-beginning

## Setup schema-registry

Copy **install-schema-registry.sh**, and **schema-registry.json** to /home/meosossu/rcclapps/confluent-components

Add following in env if needed:

One timeout is for schema to start, another one is for fetching new added schema:

“SCHEMA\_REGISTRY\_KAFKASTORE\_INIT\_TIMEOUT\_MS” :”120000”,

“SCHEMA\_REGISTRY\_KAFKASTORE\_TIMEOUT\_MS”:”120000”

“SCHEMA\_REGISTRY \_LOG4J\_ROOT\_LOGLEVEL” DEBUG ?

Upadated schema-registry.json (removed the

{

"id": "/confluent/schema-registry",

"instances": 2,

"cpus": 2,

"mem": 2048,

"disk": 0,

"gpus": 0,

"constraints": [],

"fetch": [],

"storeUrls": [],

"backoffSeconds": 1,

"backoffFactor": 1.15,

"maxLaunchDelaySeconds": 3600,

"container": {

"type": "DOCKER",

"volumes": [],

"docker": {

"image": "master.mesos:5000/confluentinc/cp-schema-registry:3.2.0",

"network": "BRIDGE",

"portMappings": [

{

"containerPort": 8081,

"hostPort": 10031,

"servicePort": 10131,

"protocol": "tcp",

"name": "schema-registry",

"labels": {

"VIP\_0": "schema-registry:8081"

}

}

],

"privileged": false,

"parameters": [],

"forcePullImage": true

}

},

"healthChecks": [

{

"gracePeriodSeconds": 60,

"intervalSeconds": 60,

"timeoutSeconds": 20,

"maxConsecutiveFailures": 3,

"portIndex": 0,

"path": "/",

"protocol": "HTTP",

"ignoreHttp1xx": false

}

],

"readinessChecks": [],

"dependencies": [],

"upgradeStrategy": {

"minimumHealthCapacity": 1,

"maximumOverCapacity": 1

},

"unreachableStrategy": {

"inactiveAfterSeconds": 300,

"expungeAfterSeconds": 600

},

"killSelection": "YOUNGEST\_FIRST",

"acceptedResourceRoles": [

"\*"

],

"requirePorts": false,

"labels": {

"DCOS\_PACKAGE\_RELEASE": "6",

"DCOS\_SERVICE\_SCHEME": "http",

"DCOS\_PACKAGE\_SOURCE": "http://master.mesos:8082/repo",

"HAPROXY\_GROUP": "external",

"DCOS\_PACKAGE\_REGISTRY\_VERSION": "3.0",

"DCOS\_SERVICE\_NAME": "schema-registry",

"DCOS\_SERVICE\_PORT\_INDEX": "0",

"HAPROXY\_0\_HTTP\_BACKEND\_PROXYPASS\_PATH": "/schema-registry",

"HAPROXY\_0\_PATH": "/schema-registry",

"DCOS\_PACKAGE\_VERSION": "0.9.9-3.2.0",

"DCOS\_PACKAGE\_NAME": "confluent-schema-registry",

"DCOS\_PACKAGE\_IS\_FRAMEWORK": "false",

"HAPROXY\_0\_VHOST": "10.16.6.44,10.16.6.92"

},

"env": {

"SCHEMA\_REGISTRY\_SCHEMA\_REGISTRY\_ZK\_NAMESPACE": "dcos-service-confluent-kafka/schema-registry",

"SCHEMA\_REGISTRY\_HEAP\_OPTS": "-Xmx256M",

"SCHEMA\_REGISTRY\_KAFKASTORE\_CONNECTION\_URL": "master.mesos:2181/dcos-service-confluent-kafka",

"SCHEMA\_REGISTRY\_LISTENERS": "http://0.0.0.0:10031,http://0.0.0.0:8081"

}

}

\* pay attention to the line of "SCHEMA\_REGISTRY\_KAFKASTORE\_CONNECTION\_URL": "master.mesos:2181/dcos-service-confluent-kafka",

++ DONOT forgot to modify in .json with new ipAddrs and Hostname:

**HAPROXY\_0\_VHOST ?**

Run ./install-schema-registry.sh

Test

<http://10.165.105.158:10131/subjects/>

<http://10.165.105.159:10131/subjects/>

## Setup rest-proxy

Copy **install-rest-proxy.sh**, and **rest-proxy.json** to /home/mesossu/rcclapps/confluent-components

++ DONOT forgot to modify in **rest-proxy.json** with new public nodes’ ipAddrs and Hostname:

**HAPROXY\_0\_VHOST ?**

Run ./install-rest-proxy.sh

Test:

<http://10.165.105.158/kafka-api/topics>

## Setup control-center (this one is currently removed from mesos because of multiple instance bug)

Copy **install-control-center.sh**, and **control-center.json** to /home/mesossu/rcclapps/confluent-components

++ DONOT forgot to modify in .json with new ipAddrs and Hostname:

**HAPROXY\_0\_VHOST ?**

Run ./install-control-center.sh

Test:

<http://10.165.105.158:10133>

**Additional work:**

Goto <https://confluence.rccl.com/pages/viewpage.action?spaceKey=MID&title=Docker+Registry+Certs>

Update the page with new .crt file.

Update Joel, provide him the .crt, maybe provide him the ‘wget’ command which allows him to get it and let him to push to bitbucket.

Goto <https://confluence.rccl.com/pages/viewpage.action?spaceKey=MID&title=Environments> , modify 2 sections