v1.11

OPERATIONS

MESOSPHERE

IF WE MIGRATE THE ENTERPRISE TO CLOUD, USE ANALYTICS, IOT AND DEVOPS FOR ALL SERVICES,
THE ENTIRE COMPANY CAN BE MANAGED BY ONE MONKEY.
PLUS A SECOND MONKEY TO LOOK AT THE POWERPOINT SLIDES FROM THE FIRST MONKEY.
Dilbert.com DilbertCartoonist@gmail.com
1-18-10 © 2010 Scott Adams, Inc./Dist. by UFS, Inc.

AGENDA

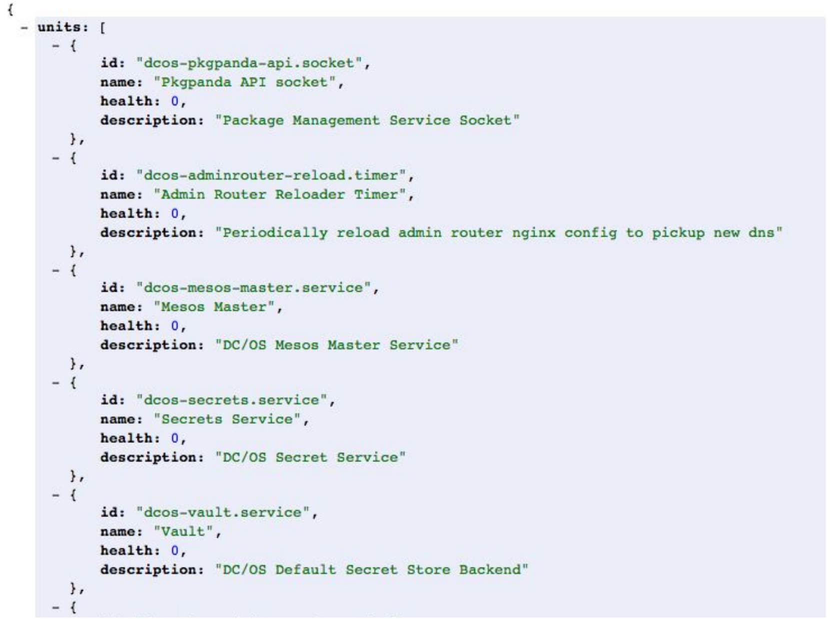
1. DCOS System Component Health
2. Log Management
3. Monitoring and Metrics
4. Failure Handling
5. Upgrades
6. Recovery
7. Production checklist

Operations

DCOS SYSTEM COMPONENT HEALTH

DCOS SYSTEM MONITORING

- Monitor the health of your cluster components from the DC/OS UI or API.
- Monitors health of the DC/OS systemd Units running on DCOS Masters and Agents with a dcos-3dt services.
    - 0 - Healthy
    - 1 - Unhealthy for one of more nodes
- API Endpoints - Use GET
    - `http://<master-ip>/system/health/v1/units`
    - `http://<master-ip>/system/health/v1/nodes`
    - `http://<master-ip>/system/health/v1/report`
    -



```
{
  - units: [
    - {
          id: "dcos-pkgpanda-api.socket",
          name: "Pkgpanda API socket",
          health: 0,
          description: "Package Management Service Socket"
      },
    - {
          id: "dcos-adminrouter-reload.timer",
          name: "Admin Router Reloader Timer",
          health: 0,
          description: "Periodically reload admin router nginx config to pickup new dns"
      },
    - {
          id: "dcos-mesos-master.service",
          name: "Mesos Master",
          health: 0,
          description: "DC/OS Mesos Master Service"
      },
    - {
          id: "dcos-secrets.service",
          name: "Secrets Service",
          health: 0,
          description: "DC/OS Secret Service"
      },
    - {
          id: "dcos-vault.service",
          name: "Vault",
          health: 0,
          description: "DC/OS Default Secret Store Backend"
      },
    - {
```

Operations

LOG MANAGEMENT

LOGGING

- Internal - Masters, Agents
- Containers - Executors, Tasks
- External - Frameworks (Marathon), Zookeeper

DC/OS LOGS

- DC/OS logs to journald
- Two main types of DC/OS Logs
  - **DC/OS System Logs** - The logs for all DC/OS Core components running on Cluster such as dcos-adminrouter, dcos-marathon, dcos-mesos-master, dcos-mesos-slave etc.
  - **DC/OS Service Logs** - Mesos STDERR and STDOUT logs for a service.
- Log rotation is configured automatically during installation.

DC/OS NODE AND SERVICE LOGS

- With DC/OS, you can inspect logs locally wherever DC/OS Cli is installed
- Traffic is pulled over HTTP / HTTPS
- Examples:

```
    $ dcos node log --leader --lines 60
    $ dcos node log --leader --follow
```

Return logs for a specific Service on DC/OS

```
    $ dcos service log marathon
    $ dcos service log hdfs --lines 25
```

DC/OS TASK LOGS

Return all active tasks

```
$ dcos task
```

Remember or Copy “Name” value for task you are interested in

Return all task output from the hello-world Task

```
$ dcos task log hello-world
```

Return all task output from the node-0 task which is actually a cassandra node.

```
$ dcos task log node-0
```

EXTERNAL INTEGRATION

- Send system and application logs from a DC/OS cluster to ELK or Splunk
- Requires connectivity to ELK/Splunk via HTTP or HTTPS (HTTPS only for ELK)
- High Level Steps
    - Install appropriate forwarder (logstash or universal forwarder) on all nodes in the cluster.
    - ELK - Create a logstash.conf for your ELK configuration
    - Create a bash script that runs journalctl on all DC/OS-units
        - ELK - Enable the service.
        - Splunk - Add the script as an input to the forwarder
    - Agent Nodes
        - Same as Masters but will also send stdout and stderr Task logs from executors

Operations

MONITORING & METRICS

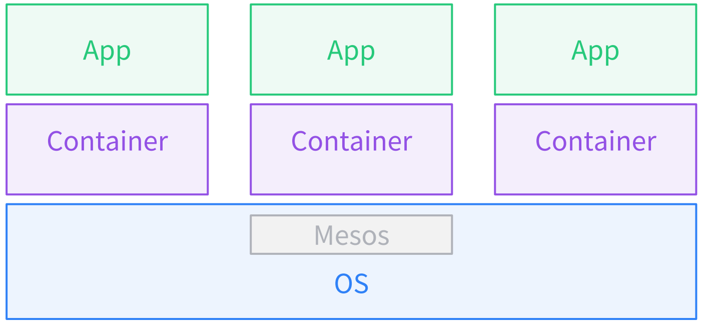
METRICS

Measurements captured to determine health and performance of cluster

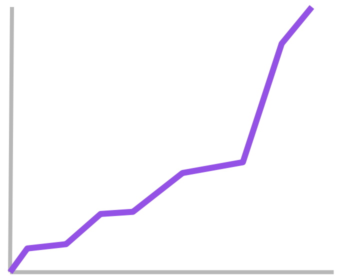
- How utilized is the cluster?
- Are resources being optimally used?
- Is the system performing better or worse over time?
- Are there bottlenecks in the system?
- What is the response time of applications?

DC/OS METRIC SOURCES

- Mesos metrics
  - Resource, frameworks, masters, agents, tasks, system, events
- Container Metrics
  - CPU, mem, disk, network
- Application Metrics
  - QPS, latency, response time, hits, active users, errors



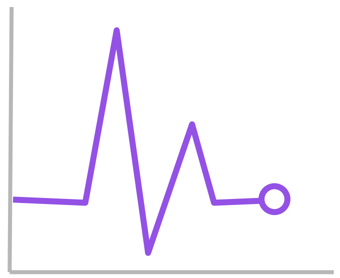
MESOS METRIC TYPES



Counters

Discrete events that are monotonically increasing.

- # of failed tasks
- # of agent registrations



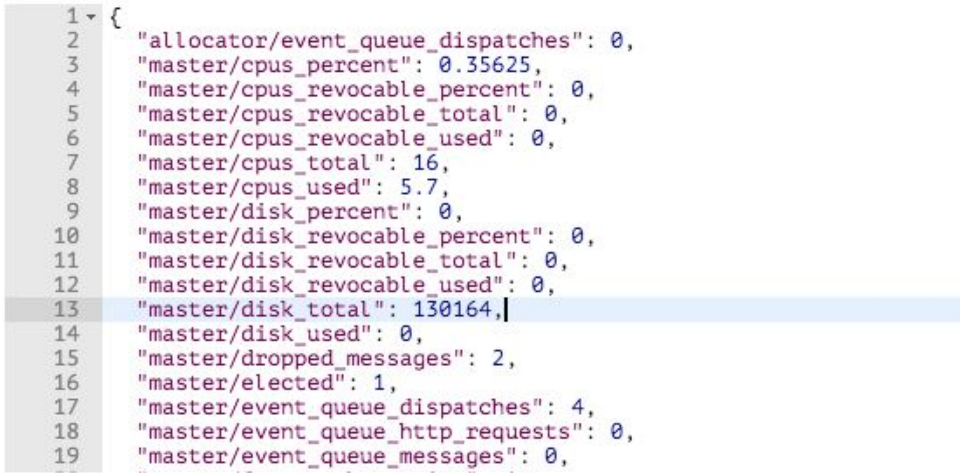
Gauges

An instantaneous sample of some magnitude.

- % of used memory in cluster
- # of connected slaves

MESOS MASTER METRICS

- Metrics for the master node are available at the following URL:
  - `http://<mesos-master-ip>/mesos/master/metrics/snapshot`
  - The response is a JSON object that contains metrics names and values as key-value pairs.
- Metric Groups:
  - Resources
  - Master
  - System
  - Slaves
  - Frameworks
  - Tasks
  - Messages
  - Event Queue
  - Registrar



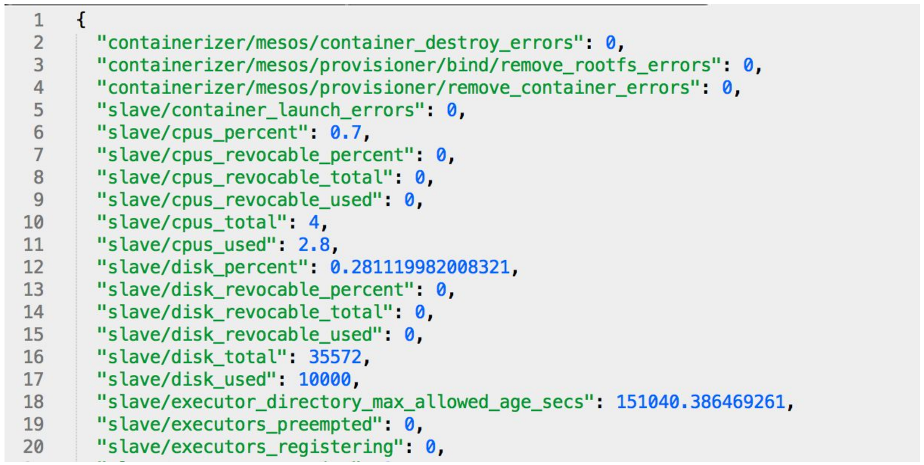
```
{
  "allocator/event_queue_dispatches": 0,
  "master/cpus_percent": 0.35625,
  "master/cpus_revocable_percent": 0,
  "master/cpus_revocable_total": 0,
  "master/cpus_revocable_used": 0,
  "master/cpus_total": 16,
  "master/cpus_used": 5.7,
  "master/disk_percent": 0,
  "master/disk_revocable_percent": 0,
  "master/disk_revocable_total": 0,
  "master/disk_revocable_used": 0,
  "master/disk_total": 130164,
  "master/disk_used": 0,
  "master/dropped_messages": 2,
  "master/elected": 1,
  "master/event_queue_dispatches": 4,
  "master/event_queue_http_requests": 0,
  "master/event_queue_messages": 0,
```

MESOS MASTER BASIC ALERTS

Metric Value	Inference
master/uptime_secs is low	The master has restarted
master/uptime_secs < 60 for sustained periods of time	The cluster has a flapping master node
master/tasks_lost is increasing rapidly	Tasks in the cluster are disappearing. Possible causes include hardware failures, bugs in one of the frameworks or bugs in Mesos
master/slaves_active is low	Slaves are having trouble connecting to the master
master/cpus_percent > 0.9 for sustained periods of time	DCOS Cluster CPU utilization is close to capacity
master/mem_percent > 0.9 for sustained periods of time	DCOS Cluster Memory utilization is close to capacity
master/disk_used & master/disk_percent	DCOS Disk space consumed by Reservations
master/elected is 0 for sustained periods of time	No Master is currently elected

MESOS AGENT METRICS

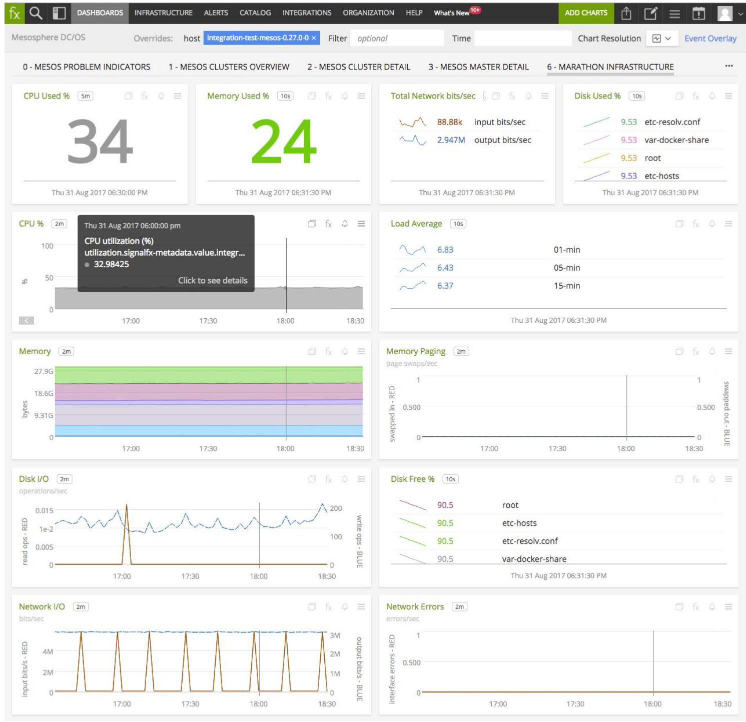
- Metrics for the agent node are available at the following URL:
  `http://<mesos-agent-ip>:5051/metrics/snapshot`
  - The response is a JSON object that contains metrics names and values as key-value pairs.
- Metric groups:
  - Resources
  - Slave
  - System
  - Executors
  - Tasks
  - Messages



```
{
  "containerizer/mesos/container_destroy_errors": 0,
  "containerizer/mesos/provisioner/bind/remove_rootfs_errors": 0,
  "containerizer/mesos/provisioner/remove_container_errors": 0,
  "slave/container_launch_errors": 0,
  "slave/cpus_percent": 0.7,
  "slave/cpus_revocable_percent": 0,
  "slave/cpus_revocable_total": 0,
  "slave/cpus_revocable_used": 0,
  "slave/cpus_total": 4,
  "slave/cpus_used": 2.8,
  "slave/disk_percent": 0.281119982008321,
  "slave/disk_revocable_percent": 0,
  "slave/disk_revocable_total": 0,
  "slave/disk_revocable_used": 0,
  "slave/disk_total": 35572,
  "slave/disk_used": 10000,
  "slave/executor_directory_max_allowed_age_secs": 151040.386469261,
  "slave/executors_preempted": 0,
  "slave/executors_registering": 0,
```

MARATHON METRICS

- Metrics for Marathon are available at the following URL:
  - `http://<marathon-ip>:8080/metrics`
  - `for DC/OS http://<master-ip>:/marathon/metrics`
- Redirect metrics to both graphite and datadog when you start the Marathon process by adding the following flag: `--reporter_graphite tcp://<graphite-server>:2003?prefix=marathon-test&interval=10`



MARATHON APP PERFORMANCE (1 OF 2)

```
$ curl <leader.mesos>/marathon/v2/apps/sleep | jq .
```

- Find the appId (sleep),“host”, and “id” (task ID) fields

```
"tasks": [
    {
      "id": "sleep.cb536c16-c6cf-11e5-a84d-0a43d276f399",
      "host": "10.0.3.226",
      "ports": [
        10466
      ],
      "startedAt": "2016-01-29T21:32:28.443Z",
      "stagedAt": "2016-01-29T21:32:27.644Z",
      "version": "2016-01-29T21:32:27.599Z",
      "slaveId": "caa0847c-3751-456f-a2fd-30feb7a1fda5-S1",
      "appId": "/sleep"
    }
  ]
```

MARATHON APP PERFORMANCE (2 OF 2)

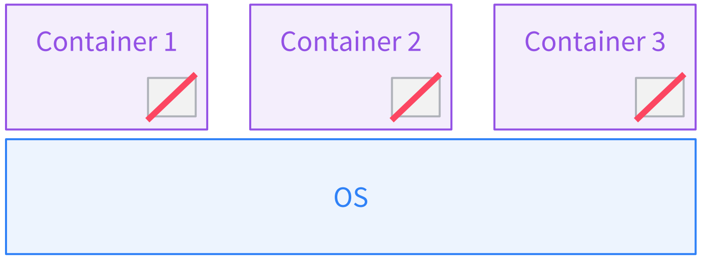
Curl the Agent host and look for the Marathon Task ID from previous step

```
$ curl http://<agent-internal-IP>:5051/monitor/statistics | jq .
```

```
{
    "executor_id": "sleep.cb536c16-c6cf-11e5-a84d-0a43d276f399",
    "executor_name": "Command Executor (Task:
sleep.cb536c16-c6cf-11e5-a84d-0a43d276f399) (Command: sh -c 'env && sleep...')",
    "framework_id": "caa0847c-3751-456f-a2fd-30feb7a1fda5-0000",
    "source": "sleep.cb536c16-c6cf-11e5-a84d-0a43d276f399",
    "statistics": {
      "cpus_limit": 0.2,
      "cpus_system_time_secs": 0,
      "cpus_user_time_secs": 0.01,
      "mem_limit_bytes": 50331648,
      "mem_rss_bytes": 200704
    }
  }
```

CONTAINER-LEVEL METRICS?

- Monitoring agent per container?
    - Not scalable
    - Increased footprint



DCOS-METRICS

Simplified config

- Container metrics (automated)
- Application metrics (statsd env vars)

Context injection

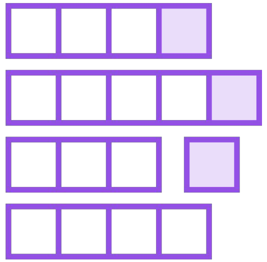
- Automated source tagging (container, agents, …)

Distributed aggregation

- Collector per node
- decoupled for faster upgrades/reconfigs

Flexible output

- Kafka, kafka consumers



INPUTS / OUTPUTS

Input: **StatsD**

- Text records: either one-per-packet or newline separated.
- Optional tagging (Datadog extension)

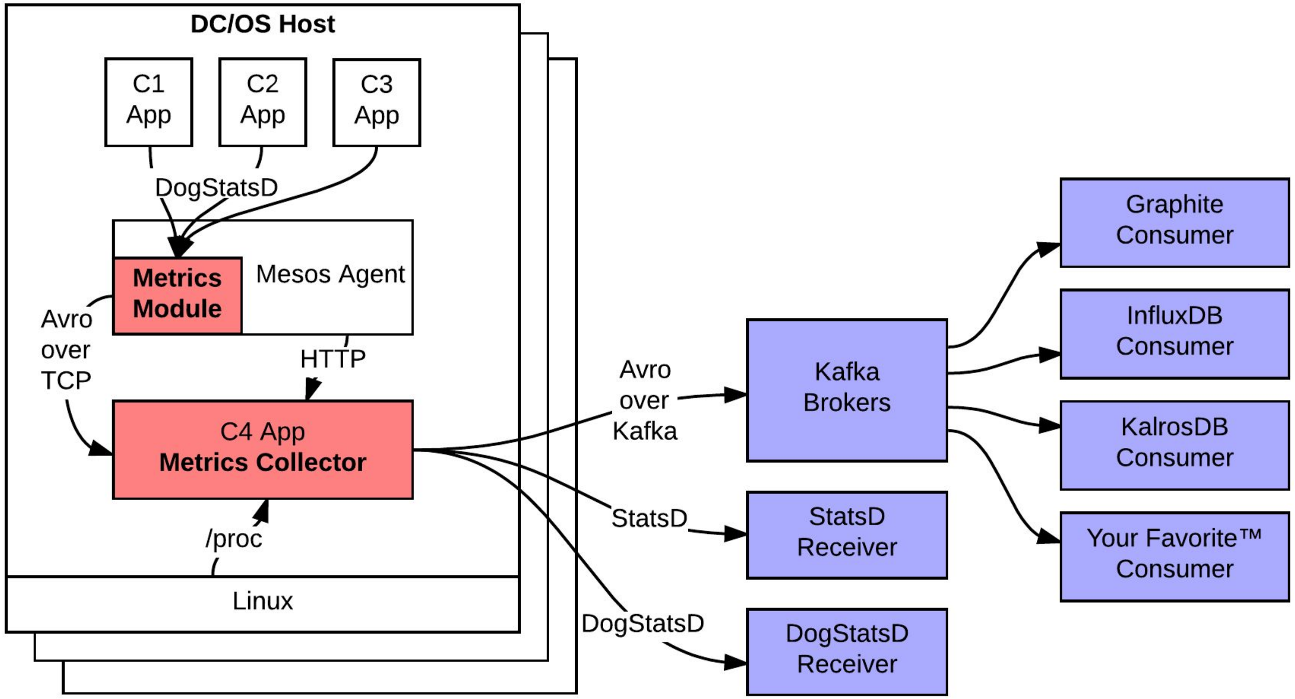
```
memory.usage_mb:5|g
frontend.query.latency_ms:46|g|#shard_id:6,section:frontpage
```

Pseudocode:

```
    if (env["STATSD_UDP_HOST"] and env["STATSD_UDP_PORT"]) {
     // 1. Open UDP socket to the endpoint
     // 2. Send StatsD-formatted metrics
    }
```

Output: **Apache Avro**

DCOS METRICS



METRICS API

Get authentication token

```
POST http://<cluster>/acs/api/v1/auth/login
{“username”: “<user>”, “password”: “<pw>”}
```

Query endpoints

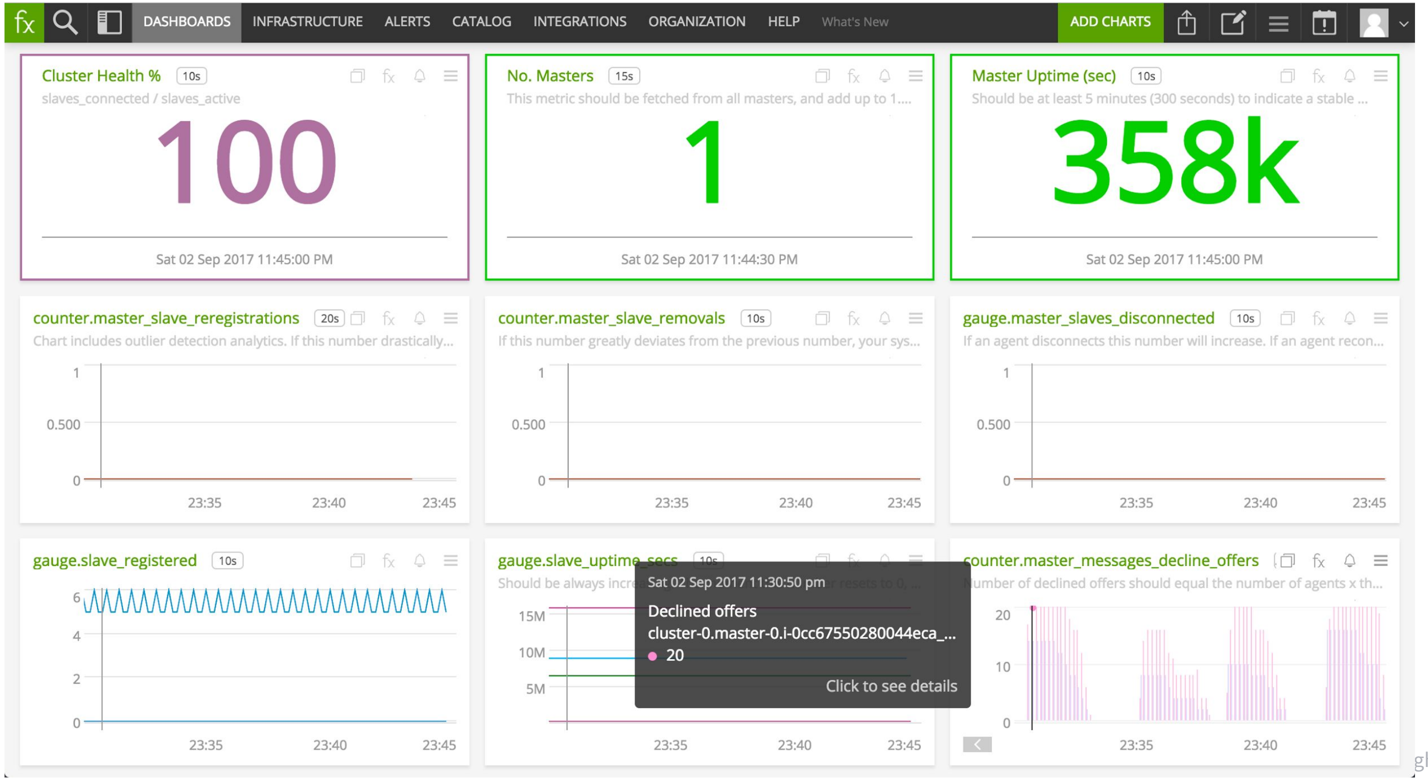
```
GET http://<cluster>/system/v1/agent/<agent_id>
/metrics/v0/<resource_path>
Accept: application/json
Authorization: token=<token_string>
```

https://docs.mesosphere.com/1.10/metrics/metrics-api/
https://docs.mesosphere.com/1.10/metrics/reference/

METRICS API

```
"datapoints": [
    {
        "name": "processes",
        "value": 209,
        "unit": "",
        "timestamp": "2017-08-31T01:00:19Z"
    },
    …
],
"dimensions": {
    "mesos_id": "a29070cd-2583-4c1a-969a-3e07d77ee665-S0",
    "hostname": "10.0.2.255"
}
```

PROBLEM INDICTATORS



Cluster Health % 10s
slaves_connected / slaves_active
100
Sat 02 Sep 2017 11:45:00 PM

No. Masters 15s
This metric should be fetched from all masters, and add up to 1....
1
Sat 02 Sep 2017 11:44:30 PM

Master Uptime (sec) 10s
Should be at least 5 minutes (300 seconds) to indicate a stable ...
358k
Sat 02 Sep 2017 11:45:00 PM

counter.master_slave_reregistrations 20s
Chart includes outlier detection analytics. If this number drastically...

counter.master_slave_removals 10s
If this number greatly deviates from the previous number, your sys...

gauge.master_slaves_disconnected 10s
If an agent disconnects this number will increase. If an agent recon...

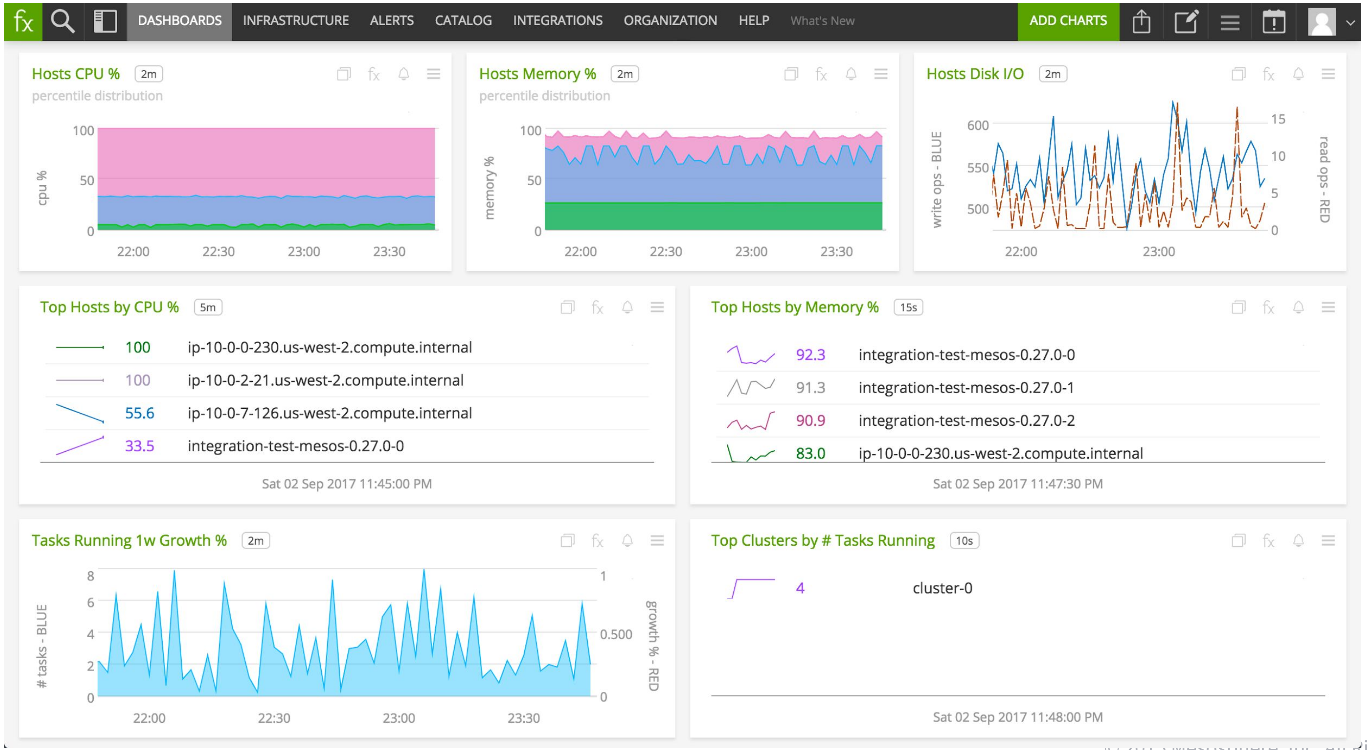
gauge.slave_registered 10s

gauge.slave_uptime_secs 10s
Should be always incre... r resets to 0, ...

Sat 02 Sep 2017 11:30:50 pm
Declined offers
cluster-0.master-0.i-0cc67550280044eca_...
● 20
Click to see details

counter.master_messages_decline_offers
umber of declined offers should equal the number of agents x th...

CLUSTER TRENDS



Top Hosts by CPU % 5m

100	ip-10-0-0-230.us-west-2.compute.internal
100	ip-10-0-2-21.us-west-2.compute.internal
55.6	ip-10-0-7-126.us-west-2.compute.internal
33.5	integration-test-mesos-0.27.0-0

Sat 02 Sep 2017 11:45:00 PM

Top Hosts by Memory % 15s

92.3	integration-test-mesos-0.27.0-0
91.3	integration-test-mesos-0.27.0-1
90.9	integration-test-mesos-0.27.0-2
83.0	ip-10-0-0-230.us-west-2.compute.internal

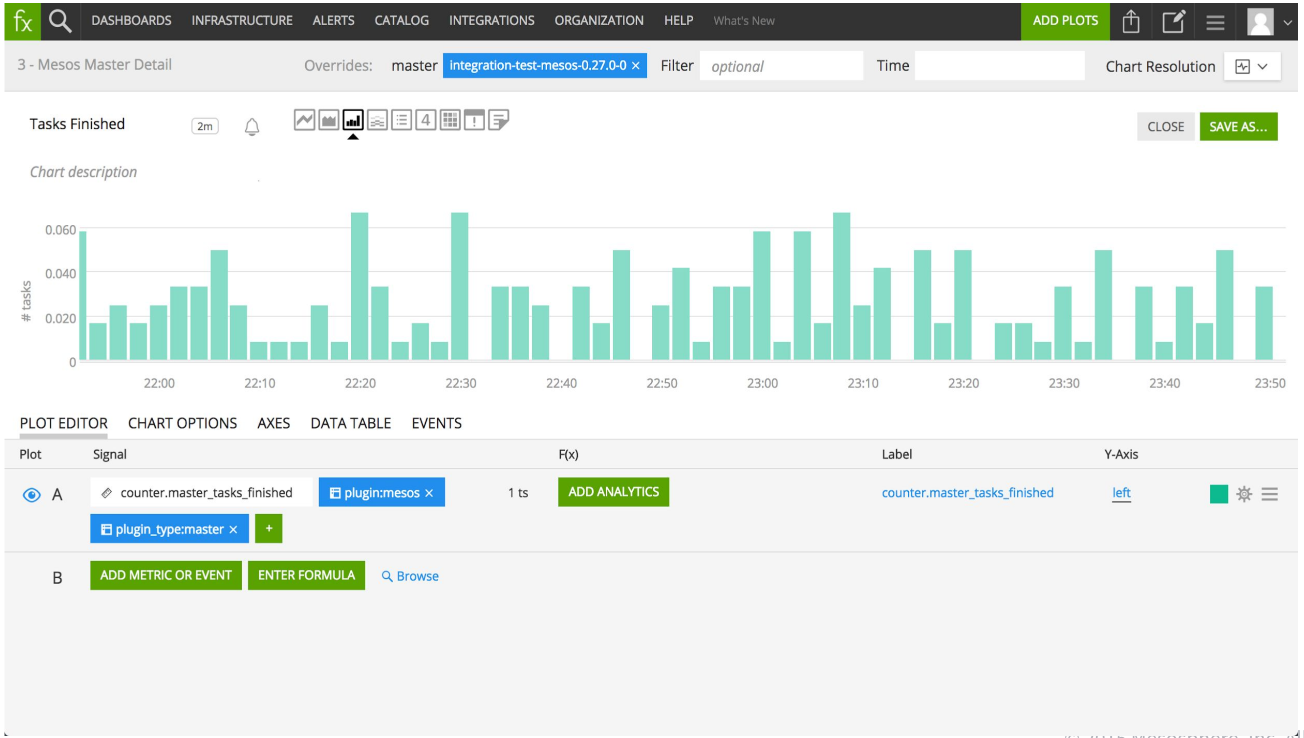
Sat 02 Sep 2017 11:47:30 PM

Top Clusters by # Tasks Running 10s

4	cluster-0

Sat 02 Sep 2017 11:48:00 PM

FILTERING BY DIMENSION



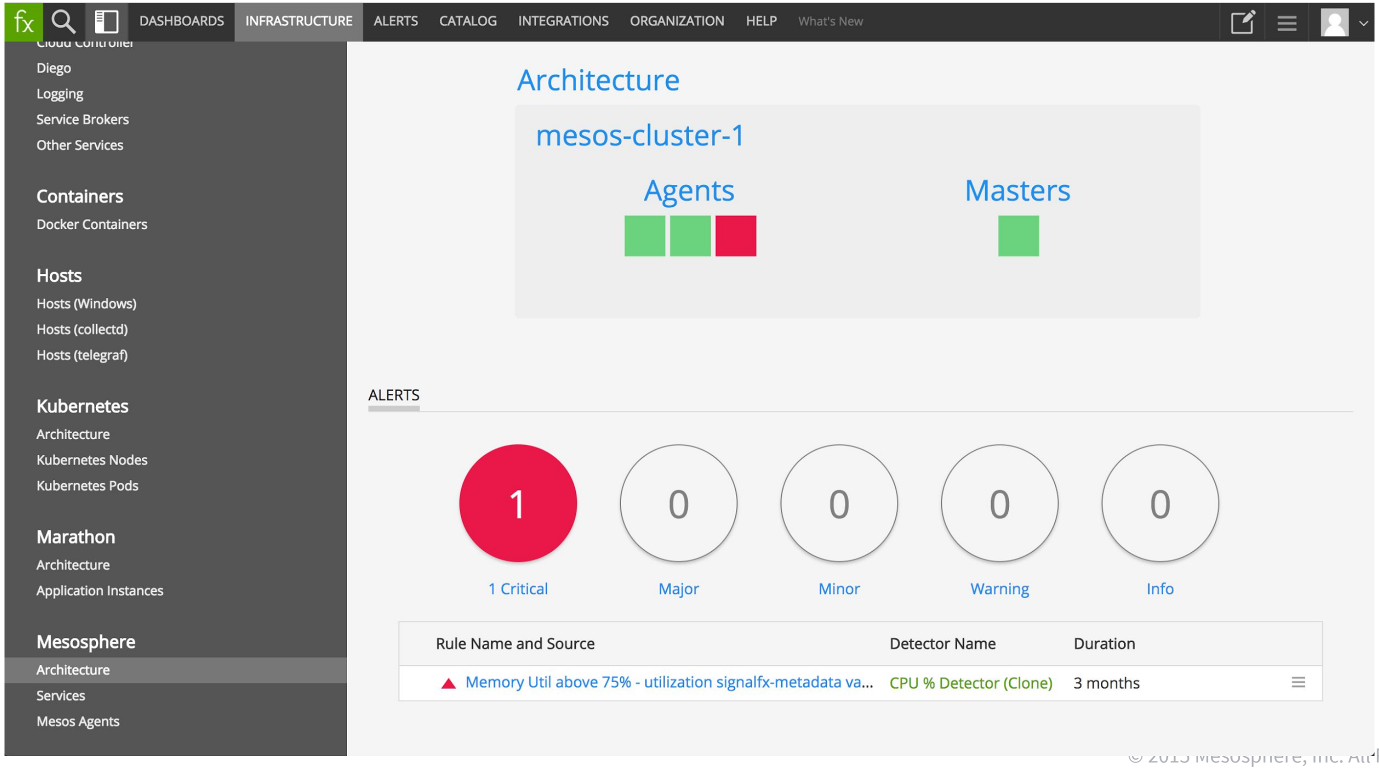
3 - Mesos Master Detail

Overrides: master integration-test-mesos-0.27.0-0 × Filter optional Time Chart Resolution

Tasks Finished 2m

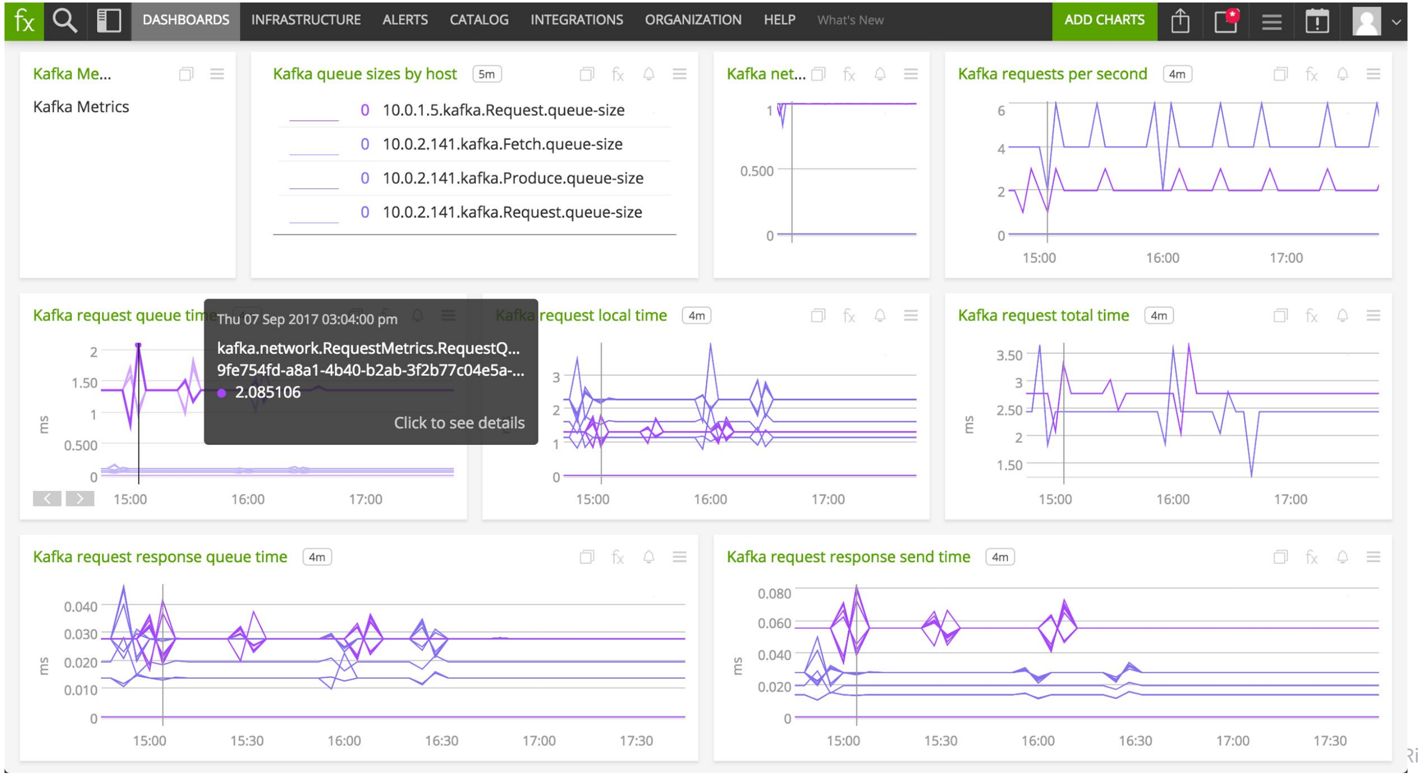
Plot	Signal		F(x)	Label	Y-Axis
A	counter.master_tasks_finished, plugin:mesos ×, plugin_type:master ×	1 ts	ADD ANALYTICS	counter.master_tasks_finished	left
B	ADD METRIC OR EVENT, ENTER FORMULA, Browse				

INFRASTRUCTURE OUTLIERS



Rule Name and Source	Detector Name	Duration
▲ Memory Util above 75% - utilization signalfx-metadata va...	CPU % Detector (Clone)	3 months

FRAMEWORK METRICS



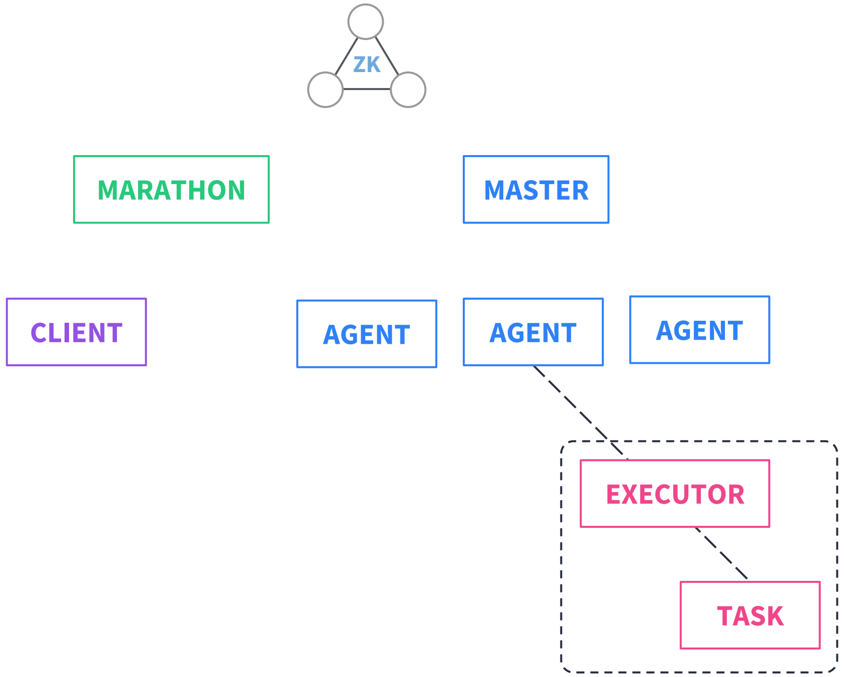
0	10.0.1.5.kafka.Request.queue-size
0	10.0.2.141.kafka.Fetch.queue-size
0	10.0.2.141.kafka.Produce.queue-size
0	10.0.2.141.kafka.Request.queue-size

Thu 07 Sep 2017 03:04:00 pm
kafka.network.RequestMetrics.RequestQ...
9fe754fd-a8a1-4b40-b2ab-3f2b77c04e5a-...
● 2.085106

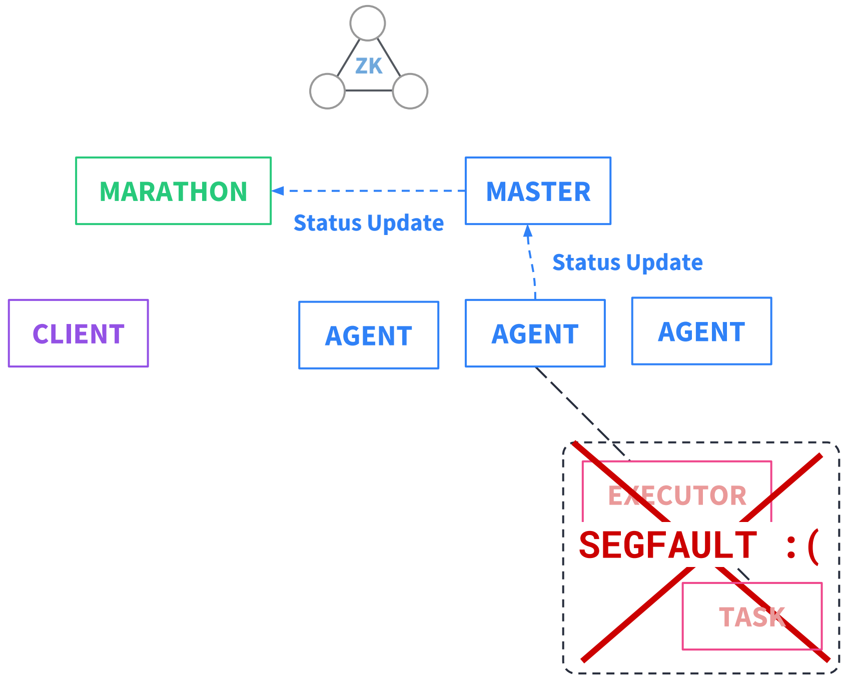
Failure Handling

MESOS TASK FAILURE

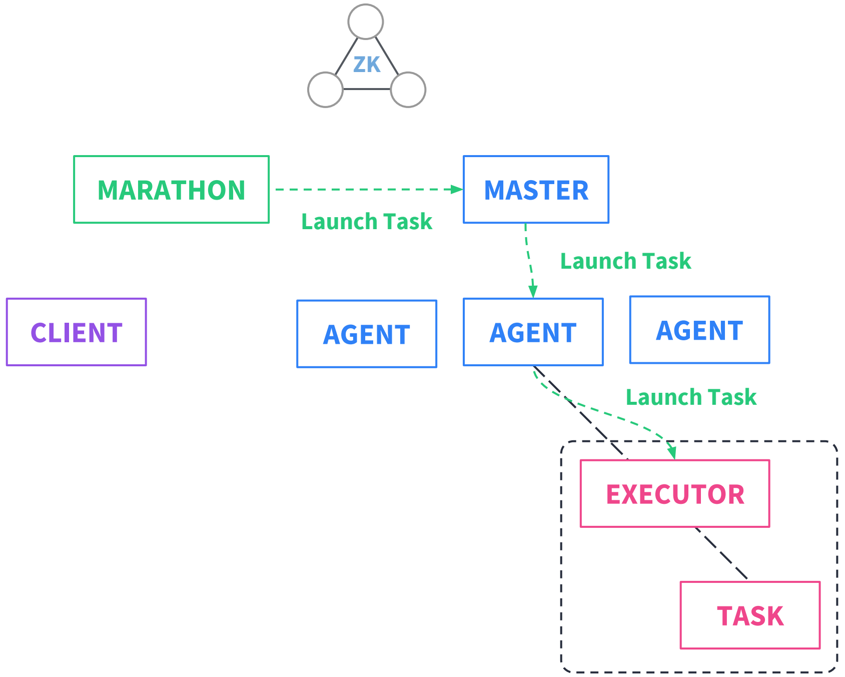
MESOS TASK FAILURE



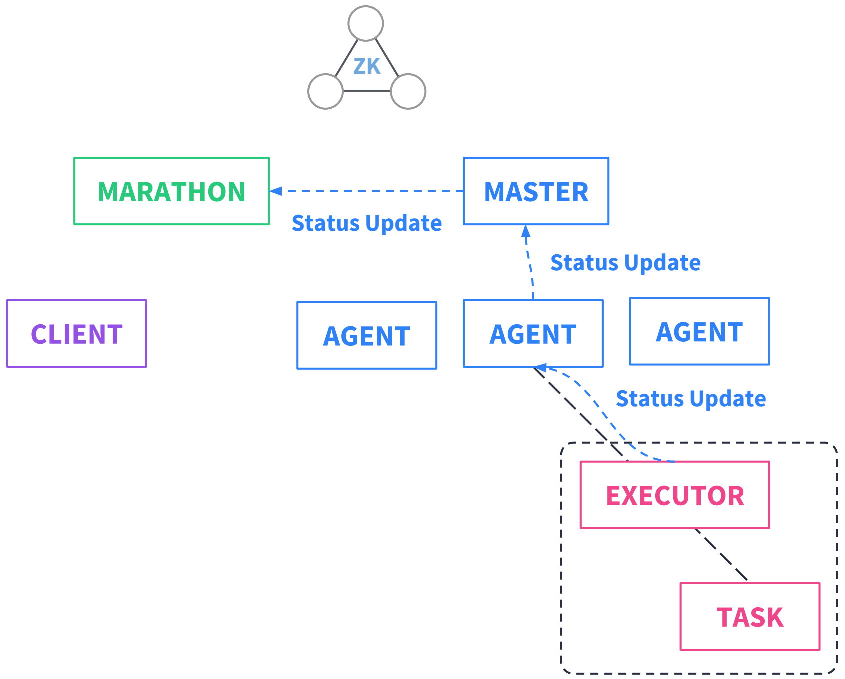
MESOS TASK FAILURE



MESOS TASK FAILURE



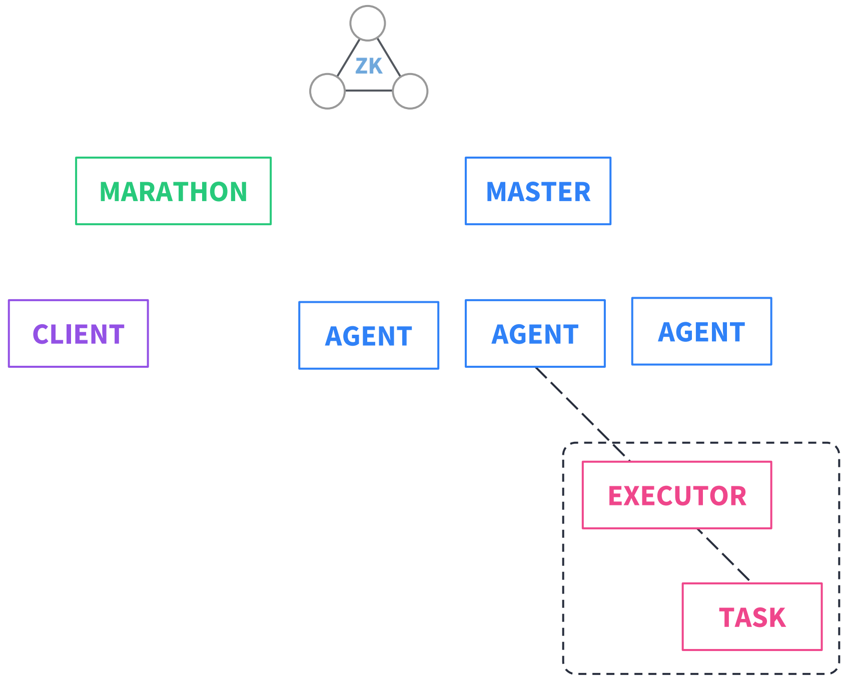
MESOS TASK FAILURE



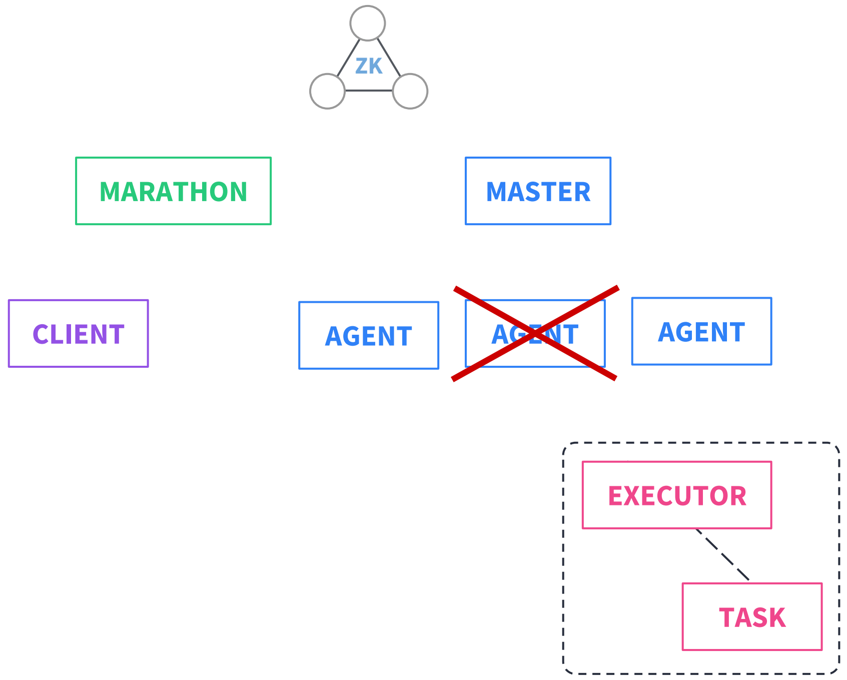
Failure Handling

MESOS AGENT FAILURE

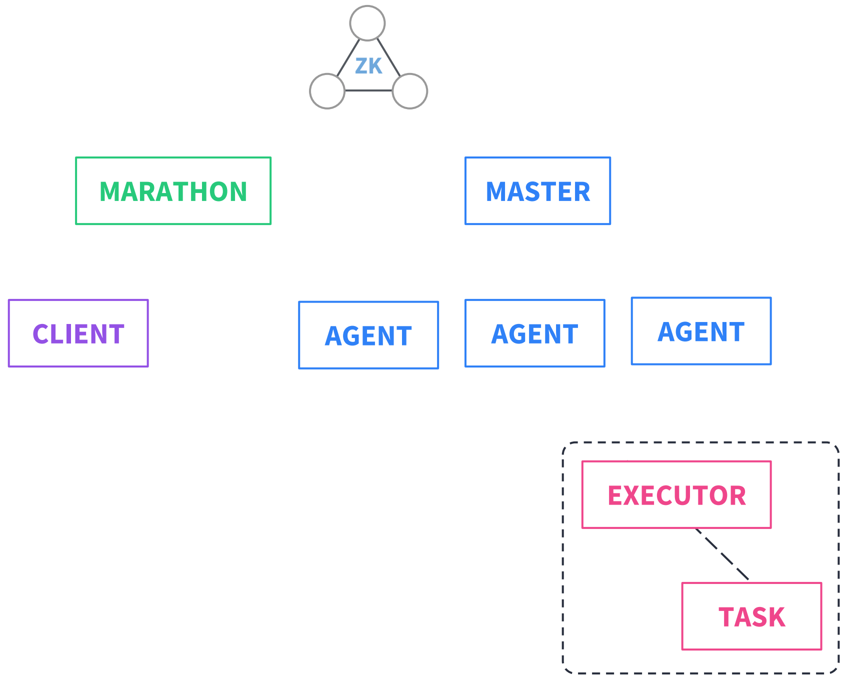
LOCAL AGENT FAILURE



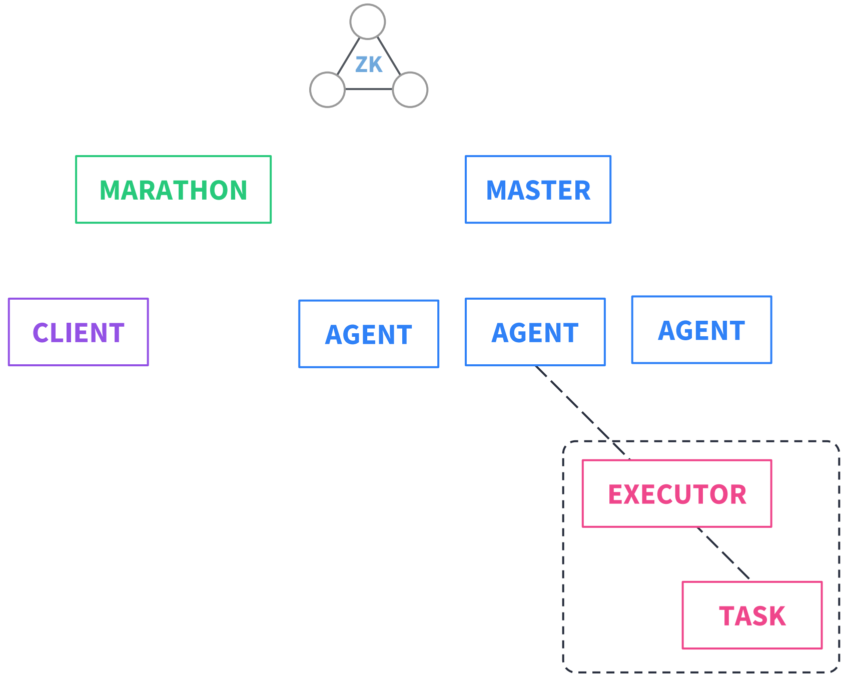
LOCAL AGENT FAILURE



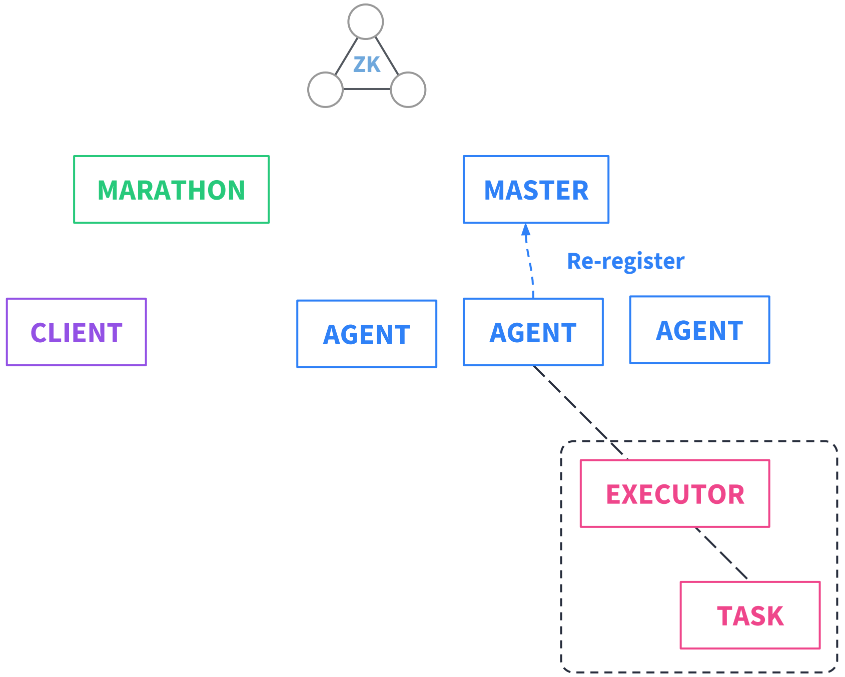
LOCAL AGENT FAILURE



LOCAL AGENT FAILURE



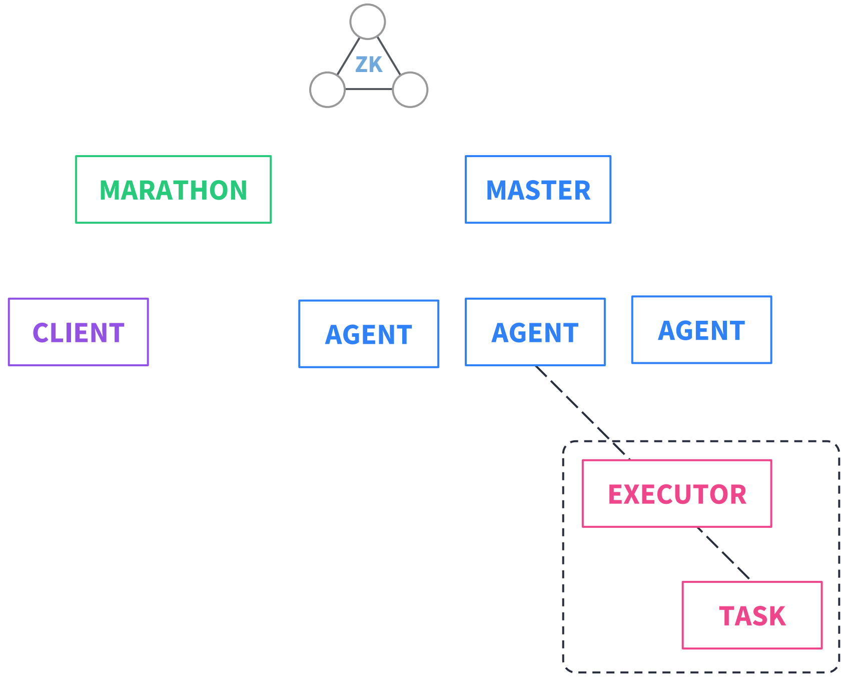
LOCAL AGENT FAILURE



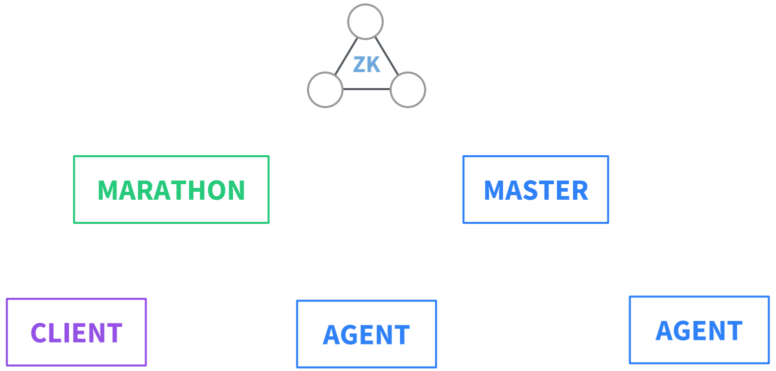
Failure Handling

MESOS HOST FAILURE

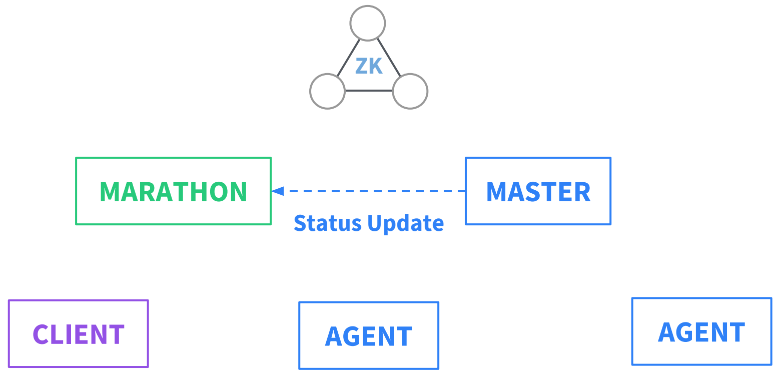
LOCAL AGENT FAILURE



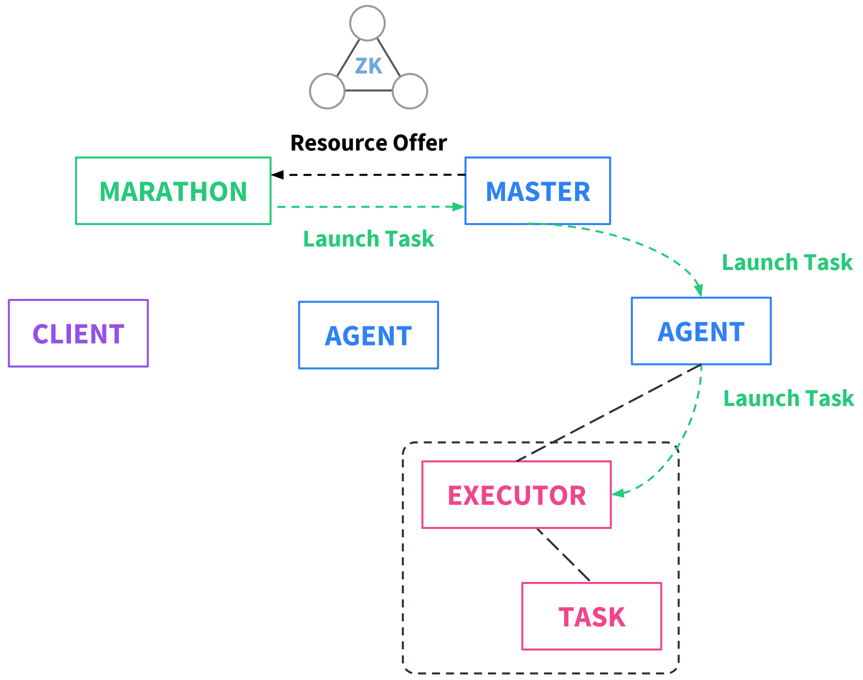
LOCAL AGENT FAILURE



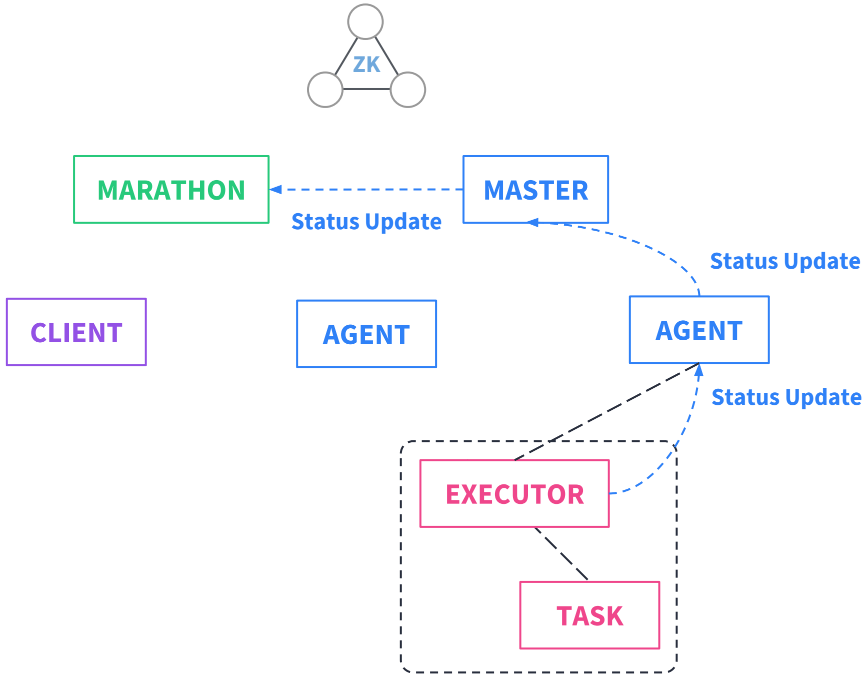
LOCAL AGENT FAILURE



MESOS TASK FAILURE



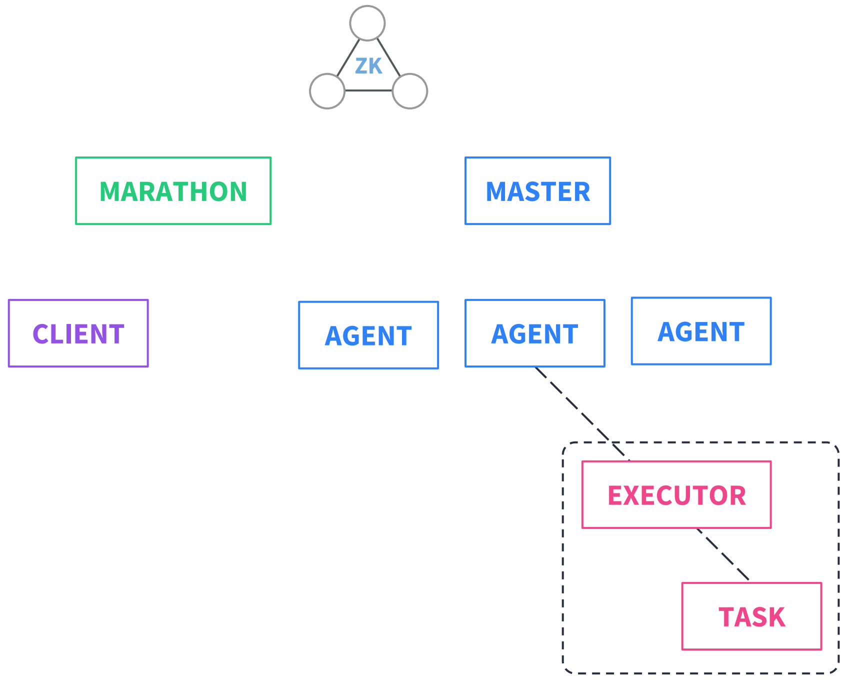
MESOS TASK FAILURE



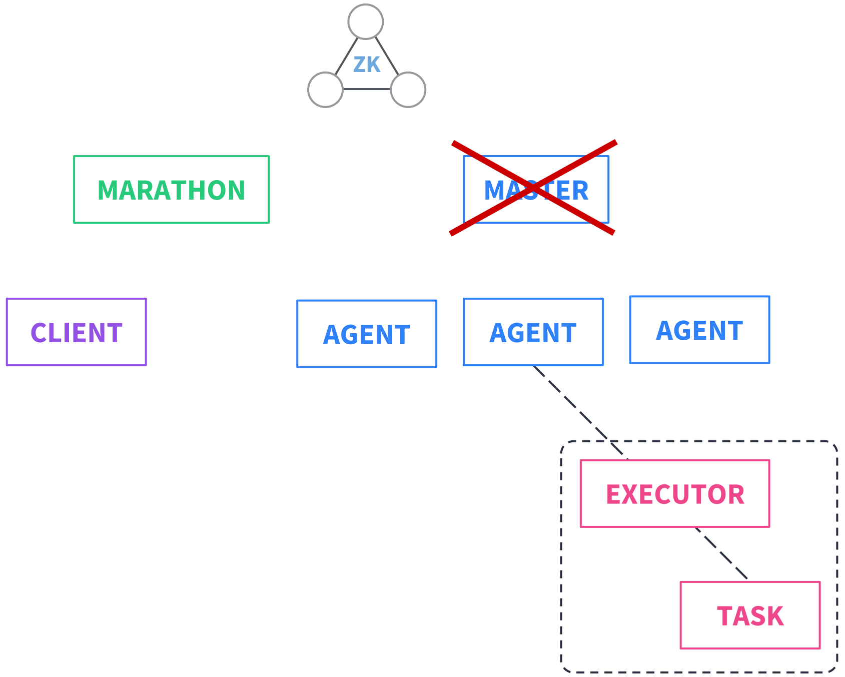
Failure Handling

MESOS MASTER FAILURE

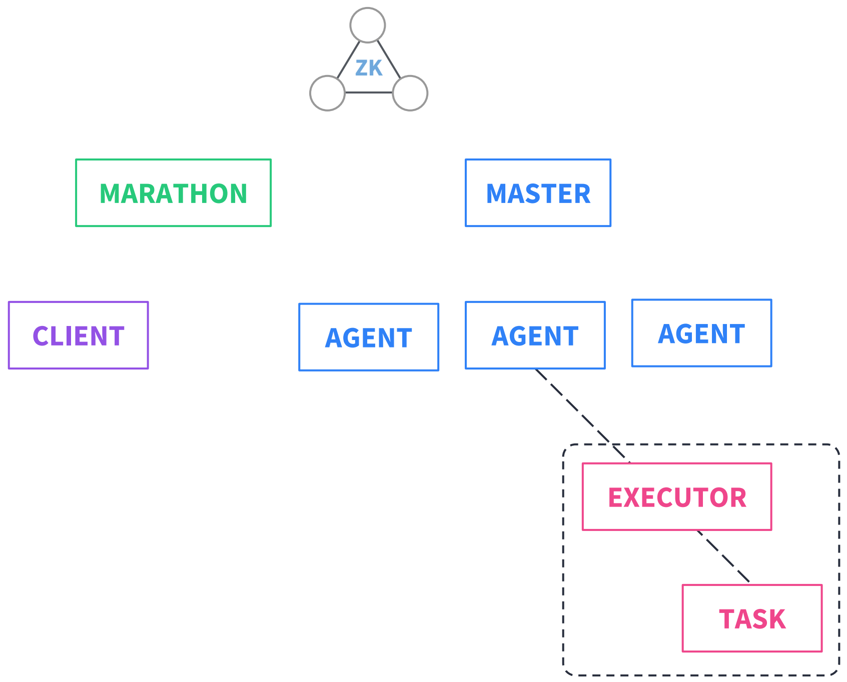
MASTER FAILURE



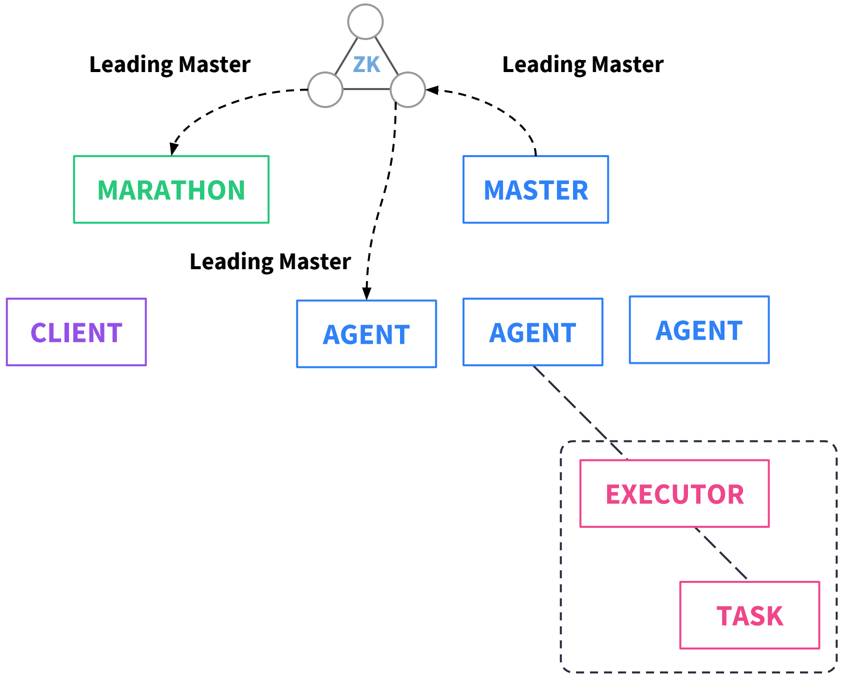
MASTER FAILURE



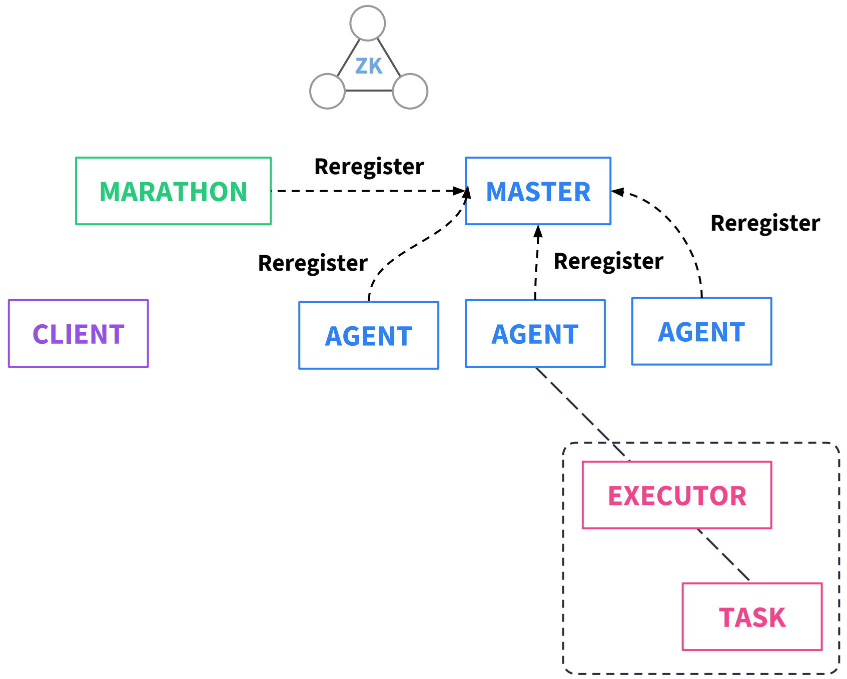
MASTER FAILURE



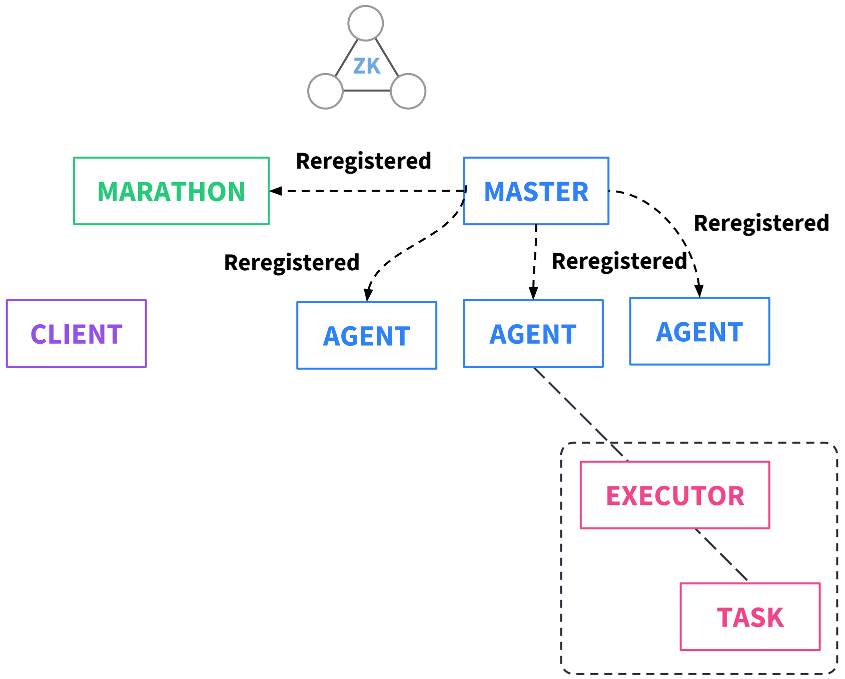
MASTER FAILURE



MASTER FAILURE



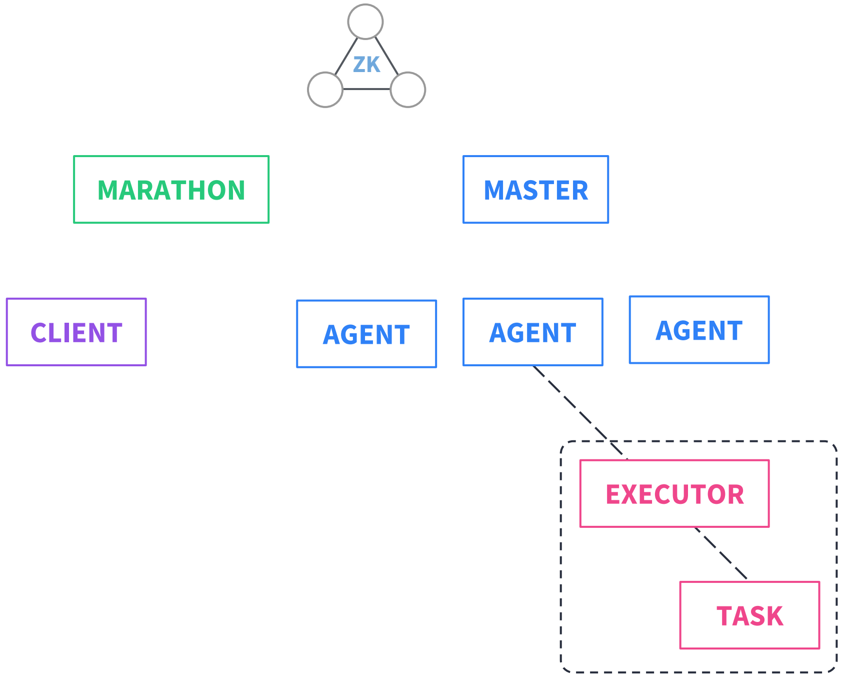
MASTER FAILURE



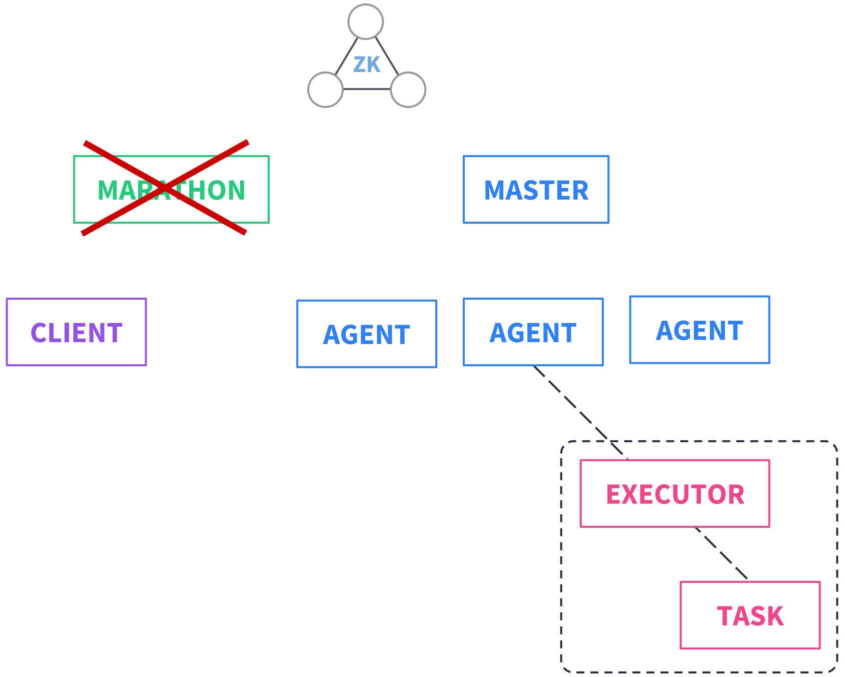
Failure Handling

SCHEDULER FAILURE

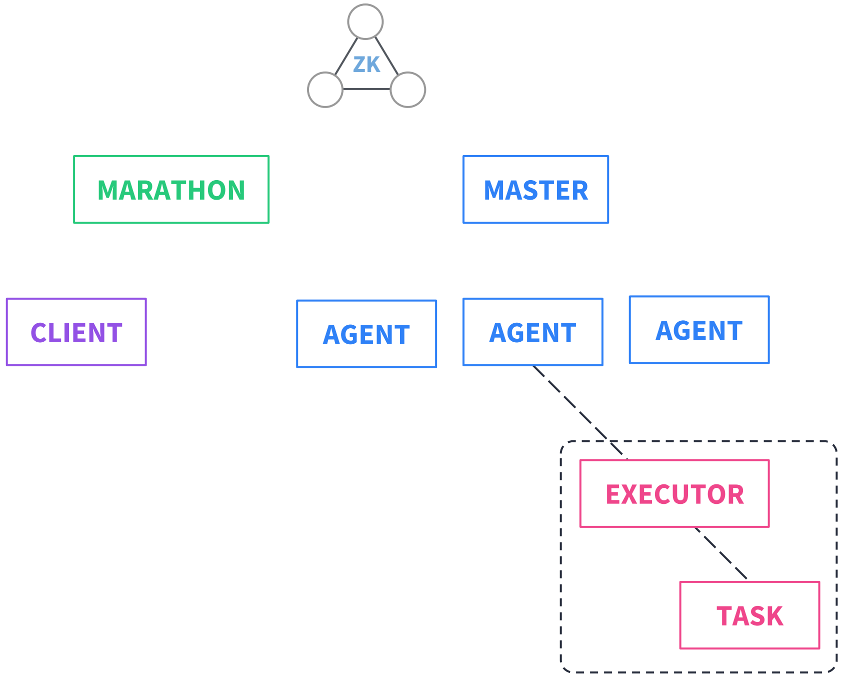
SCHEDULER FAILURE



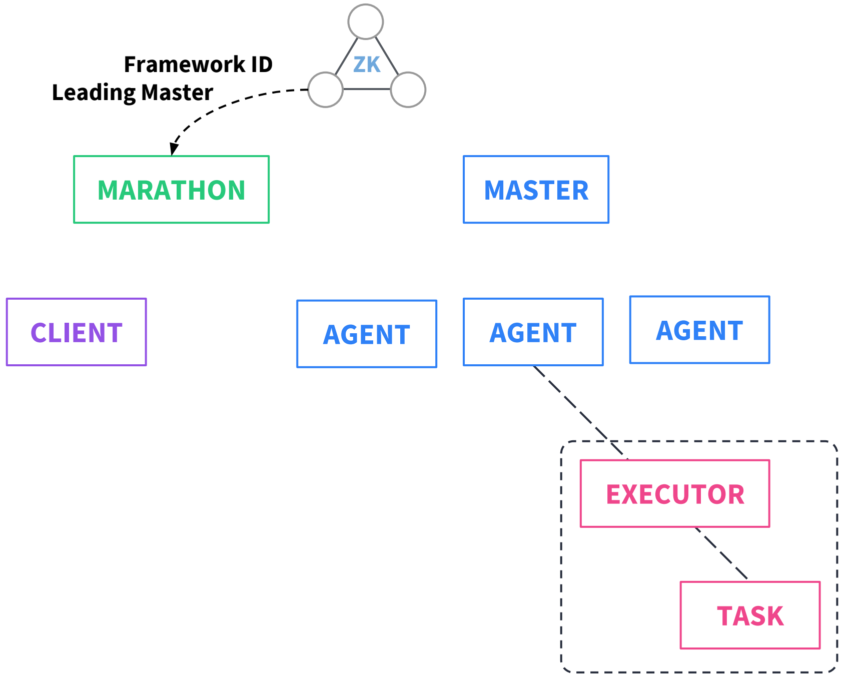
SCHEDULER FAILURE



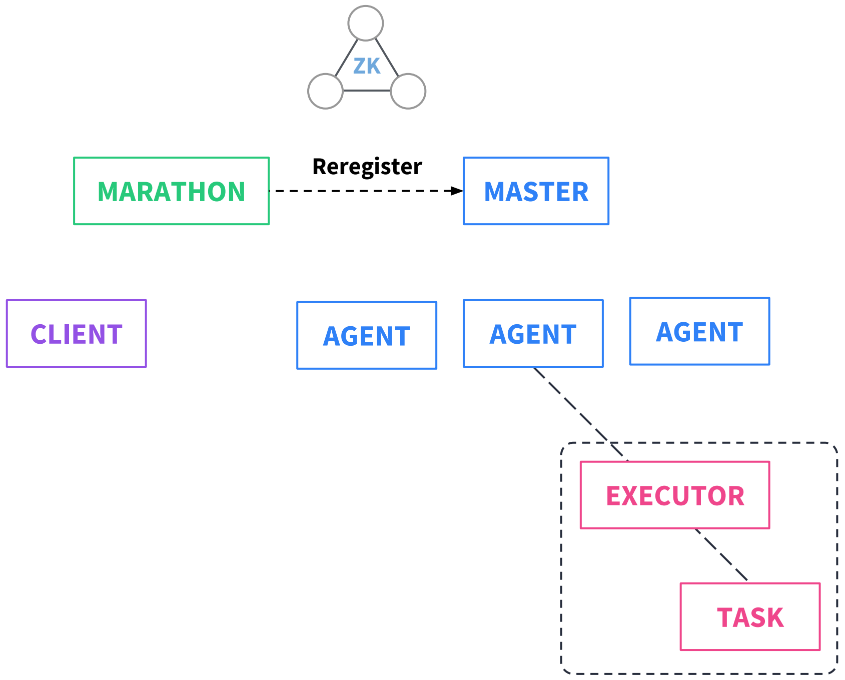
SCHEDULER FAILURE



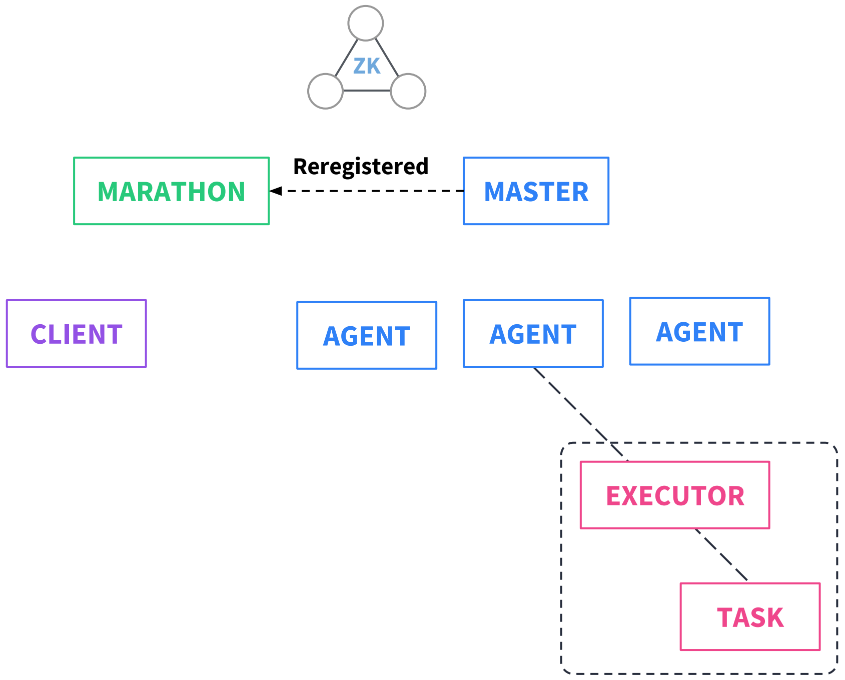
SCHEDULER FAILURE



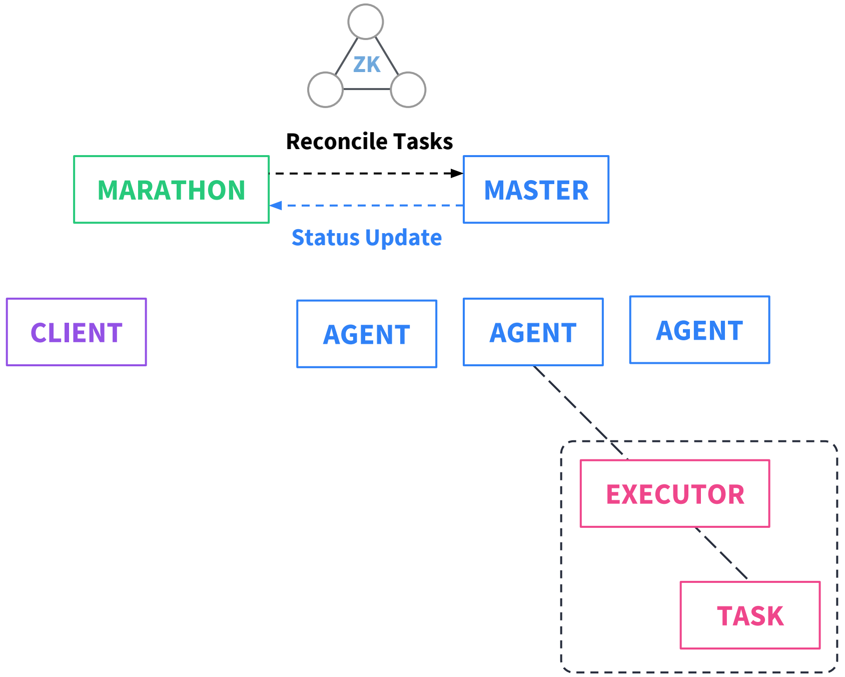
SCHEDULER FAILURE



SCHEDULER FAILURE



SCHEDULER FAILURE



Operations

UPGRADES

BACKUP AND RESTORE DC/OS

Overview:

- Available through DC/OS 1.10 EE CLI
  - `dcos package install dcos-enterprise-cli`
- Backup local Marathon instance
- Managed by a long running service
- Stored on the local filesystem of the master(s)
- Should be added to upgrade procedure

BACKUP PROCEDURE

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

Upgrades

RESTORE PROCEDURE

1. Find the backup: `dcos backup list`
2. Restore: `dcos backup restore <backup-id>`
3. View progress: `dcos backup show <backup-id>`

Upgrades

BACKUP AND RESTORE DC/OS

```
$ dcos backup list
BACKUP ID                        VERSION     STATUS             TIMESTAMP
my-backup-800a8d84-02f3-4179-80ec-cfc1236542   1.10.0 STATUS_BACKING_UP   2017-10-12T15:48:41.130

$ dcos backup show my-backup-800a8d84-02f3-4179-80ec-cfc1236542
{
  “component_status”: {
    “marathon”: {
      “status”: “STATUS_BACKING_UP”
    }
  },
  “dcos_version”: “1.10.0”,
  “id”: “my-backup-800a8d84-02f3-4179-80ec-cfc1236542”,
  “status”: “STATUS_BACKING_UP”,
  “timestamp”: “2017-10-12T15:48:41.130”
}
```

PRE AND POST UPGRADE DIAGNOSTICS

Overview:

- Node and cluster health checks
- Checks are automatically run when upgrading
- Included in the `dcos-diagnostics` bundle

Benefits:

- Helps prevent issues related to configuration drift
- Can create custom checks. Documentation
- Nagios compatibility

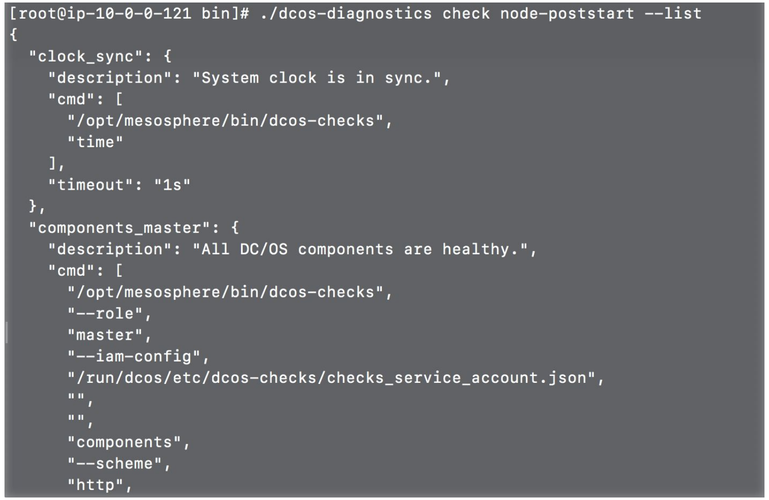
POST UPDGRADE DIAGNOSTIS CLI

Overview:

- Diagnostics CLI come pre-installed in `/opt/mesosphere/bin`

Examples:

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

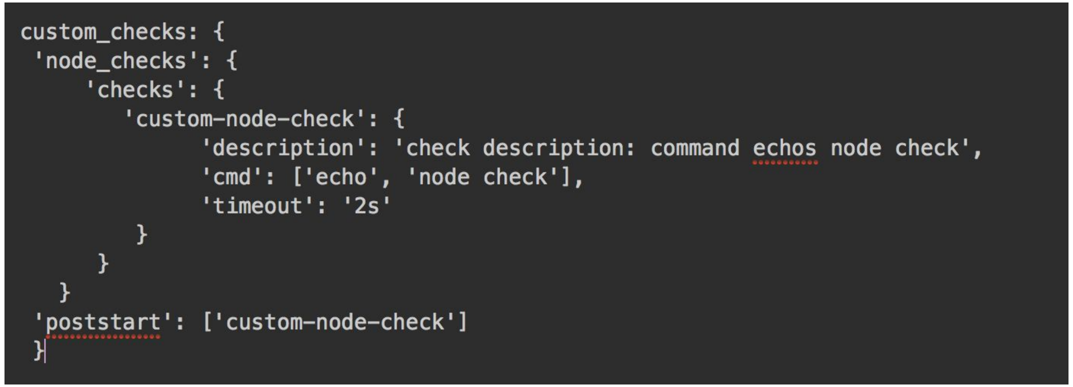


```
[root@ip-10-0-0-121 bin]# ./dcos-diagnostics check node-poststart --list
{
  "clock_sync": {
    "description": "System clock is in sync.",
    "cmd": [
      "/opt/mesosphere/bin/dcos-checks",
      "time"
    ],
    "timeout": "1s"
  },
  "components_master": {
    "description": "All DC/OS components are healthy.",
    "cmd": [
      "/opt/mesosphere/bin/dcos-checks",
      "--role",
      "master",
      "--iam-config",
      "/run/dcos/etc/dcos-checks/checks_service_account.json",
      "",
      "",
      "components",
      "--scheme",
      "http",
```

CUSTOM HEALTH CHECKS DIAGNOSTIS

Overview:

- Configured in `config.yaml`
- Two new parameters in custom-checks: “cluster-checks” and “node-checks”
- For more information: https://docs.mesosphere.com/1.10/installing/custom/configuration/configuration-parameters/#custom_checks

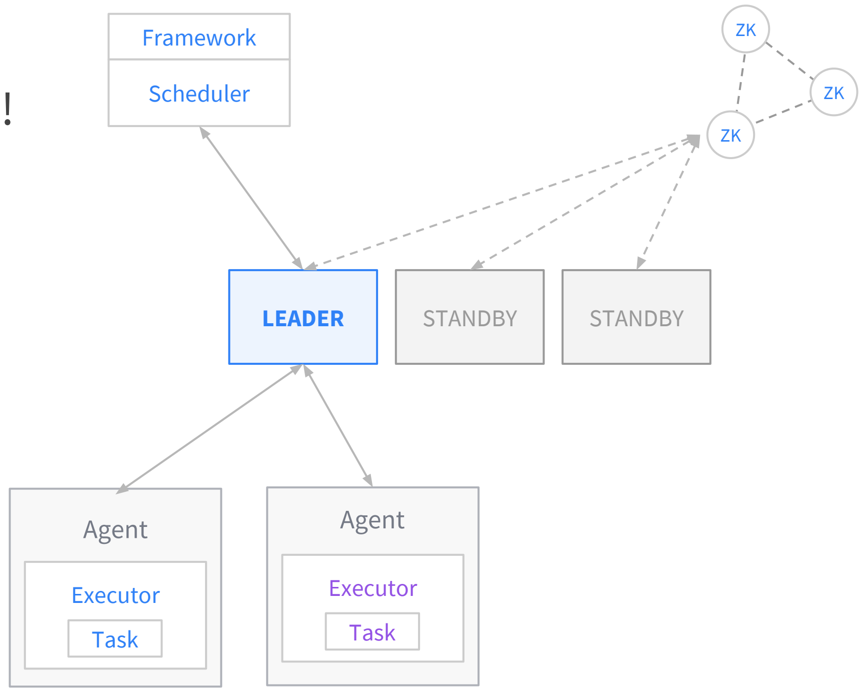


```
custom_checks: {
 'node_checks': {
     'checks': {
        'custom-node-check': {
               'description': 'check description: command echos node check',
               'cmd': ['echo', 'node check'],
               'timeout': '2s'
         }
      }
   }
 'poststart': ['custom-node-check']
 }
```

UPGRADE PROCEDURE

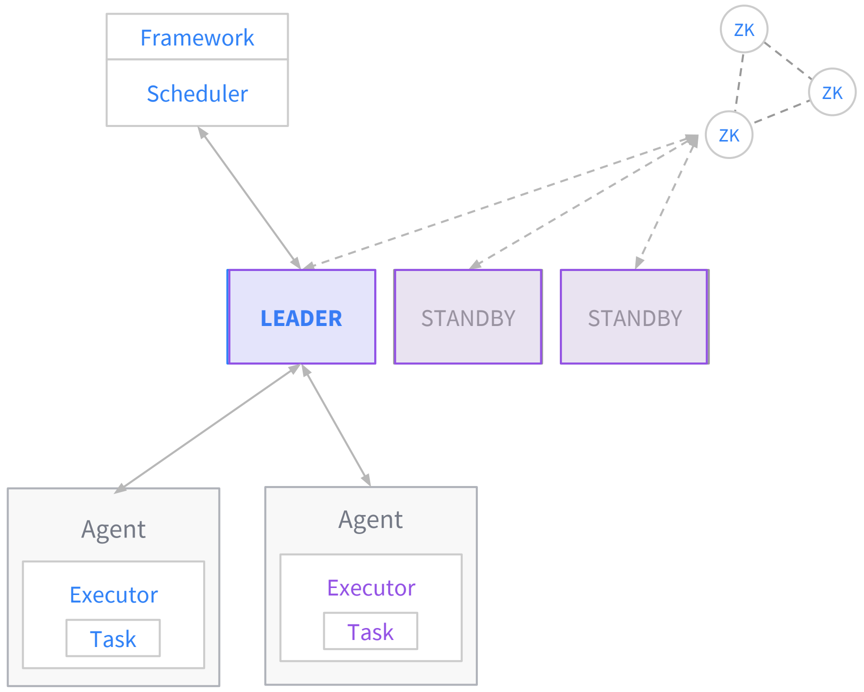
Before upgrading

1. Make sure cluster is healthy!
2. Perform backup
   a. ZK
   b. Replicated logs
   c. other state
3. Review release notes
4. Generate install bundle
   a. Validate versions



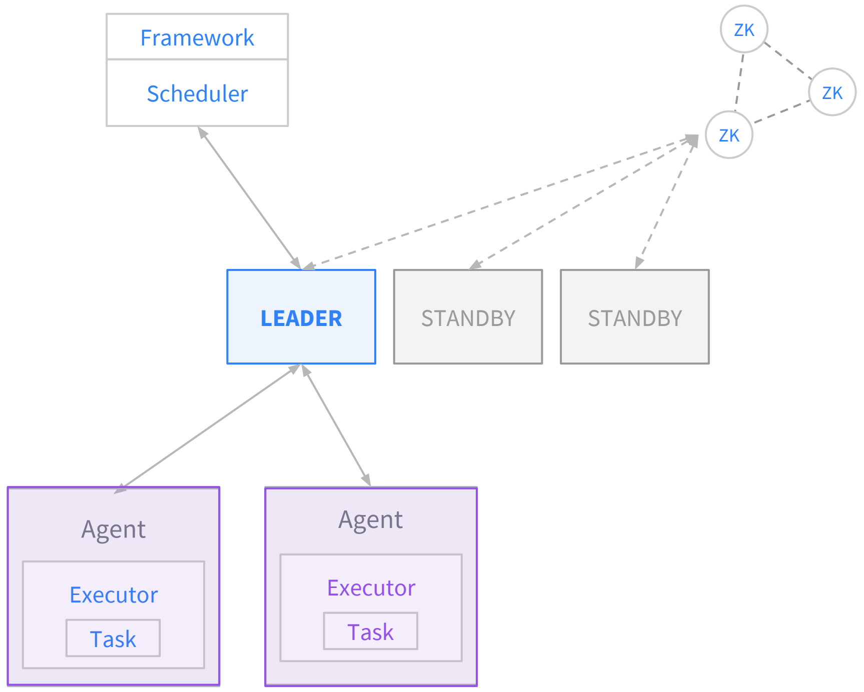
UPGRADE PROCEDURE

1. Master rolling upgrade
   a. Start with standby
   b. Uninstall DC/OS
   c. Install new DC/OS
2. Agent rolling upgrade
3. Framework upgrades



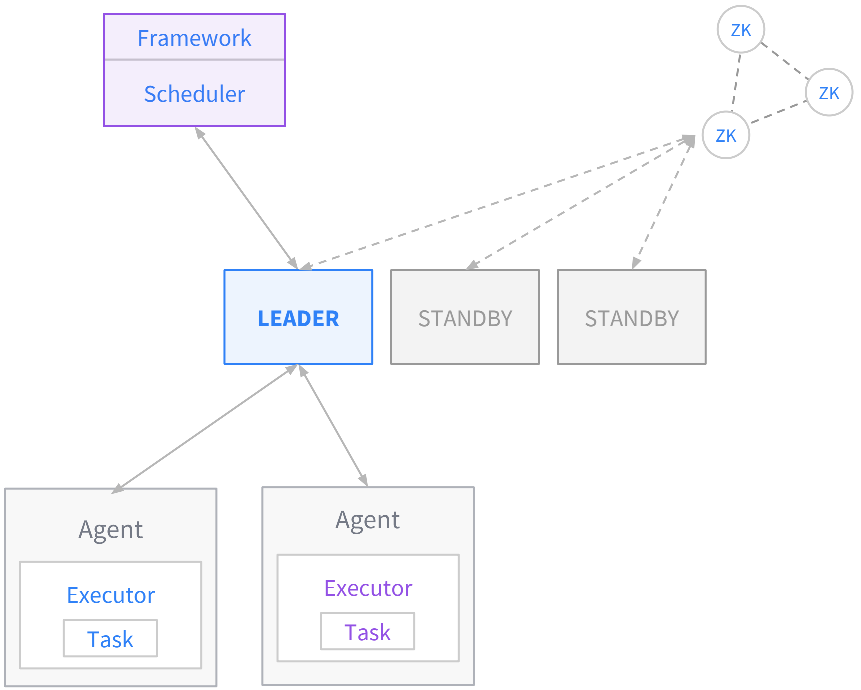
UPGRADE PROCEDURE

1. Master rolling upgrade
2. Agent rolling upgrade
   a. Uninstall DC/OS
   b. Install new DC/OS
3. Framework upgrades



UPGRADE PROCEDURE

1. Master rolling upgrade
2. Agent rolling upgrade
3. Framework upgrades
   a. Orthogonal to DC/OS
   b. Ensure changes don't affect existing apps



FUTURES (TBD)

Leverage maintenance primitives in Mesos to drain host

Upgrade management through DC/OS to perform rolling upgrades

Operations

BACKUP

WHAT TO BACKUP

- Zookeeper
- Mesos replicated log
- Stateful services state

REPLICATED LOG BACKUP AND RECOVERY

- The replicated log should be backed up periodically, such as every 24 hours
- To backup the replicated log, create a tarball from the master's work directory
- To restore the replicated log:
  - Stop the master
  - Extract the log snapshot
  - Restart the master

Operations

AGENT RECOVERY

AGENT RECOVERY

- Agent Recovery is a feature of Mesos that allows:
  - Executors/Tasks to keep running when the agent process is down
  - Restarted agent process to reconnect with running executors/tasks
- Agent recovery works by checkpointing enough information (e.g., Task Info, Executor Info, Status Updates) about the running tasks and executors to local disk.
- Once the agent and the framework(s) enable checkpointing, any subsequent agent restarts would recover the checkpointed information and reconnect with the executors/tasks.
- If the host running the agent process is rebooted, all executors/tasks are killed.

CHECKPOINTING

- Automatically enabled for all agents with Mesos 0.22.0
    - A restarted agent should re-register with the master within a timeout (currently 75s). If the agent takes longer than this timeout to re-register, the master shuts down the agent, which in turn shuts down any live executors/tasks.
- Frameworks should explicitly request checkpointing by setting `FrameworkInfo.checkpoint=True` before registering with the master.

Operations

PRODUCTION CHECKLIST

MESOS CHECKLIST

- ❏ Monitor both Masters and Agents for flapping (i.e., continuously restarting). This can be accomplished by using the `uptime` metric.
- ❏ Monitor the rate of changes in terminal task states, including TASK_FAILED, TASK_LOST, and TASK_KILLED

MESOS MASTER CHECKLIST

- ❏ Use five master instances in production. Three is sufficient for HA in staging/test
- ❏ Place masters on separate racks, if possible
- ❏ Secure the teardown endpoints to prevent accidental framework removal.

MESOS AGENT CHECKLIST

- ❏ Set agent attributes before you run anything on the cluster. Once an agent is started, changing the attributes may break recovery of running tasks in the event of a restart. See also https://issues.apache.org/jira/browse/MESOS-1739.
- ❏ Explicitly set the resources on the nodes to leave capacity for other services running there *outside* of Mesos control. For example, HDFS processes running alongside Mesos.

ZOOKEEPER CHECKLIST

- ❑ Run with security and ACLs, see the `--zk=` and `--master=` flags on the master and slaves respectively. If you do enable ACLs, they must be enabled **before** nodes are created in ZK.
- ❑ Backup ZooKeeper snapshots and log at regular intervals. -
  - ❑ Guano or zkConfig.py (Want Snapshots + Transaction Log)
- ❑ Marathon, Chronos, and other frameworks store state in ZK. The first Marathon should store state in the same ZK as Mesos master.
- ❑ Userland apps *should NOT* store state in the ZK cluster shared by Mesos and Marathon. Examples of userland apps include Storm, service discovery tools, and additional instances of Marathon and Chronos.

ZOOKEEPER CHECKLIST

- ❏ Monitor ZK's JVM metrics, such as heap usage, GC pause times, and full-collection frequency.
- ❏ Monitor ZK for: number of client connections, total number of znodes, size of znodes (min, max, avg, 99% percentile), and read/write performance metrics

MESOSPHERE

MESOS CHECKLIST

- ❏ Configure ulimit settings appropriately: 'locked memory' settings. On DC/OS, these settings are pre-configured.
  - ❏ `open files` - A value of 32000 (soft) and 262144 (hard) has been used successfully in large production deployments.
  - ❏ `max locked memory` - Increase to account for huge pages, if required. Setting to unlimited is acceptable.
- ❏ You *MAY* want to enable logging to disk with `--log_dir=/var/log/mesos`. If you enable logging to disk, perform log rotation of the files that Mesos writes (using a log rotation tool, such as `logrotate`).