Pallet

DevOps for the JVM

@tbatchelli a.k.a @disclojure

Clojure/West 2012

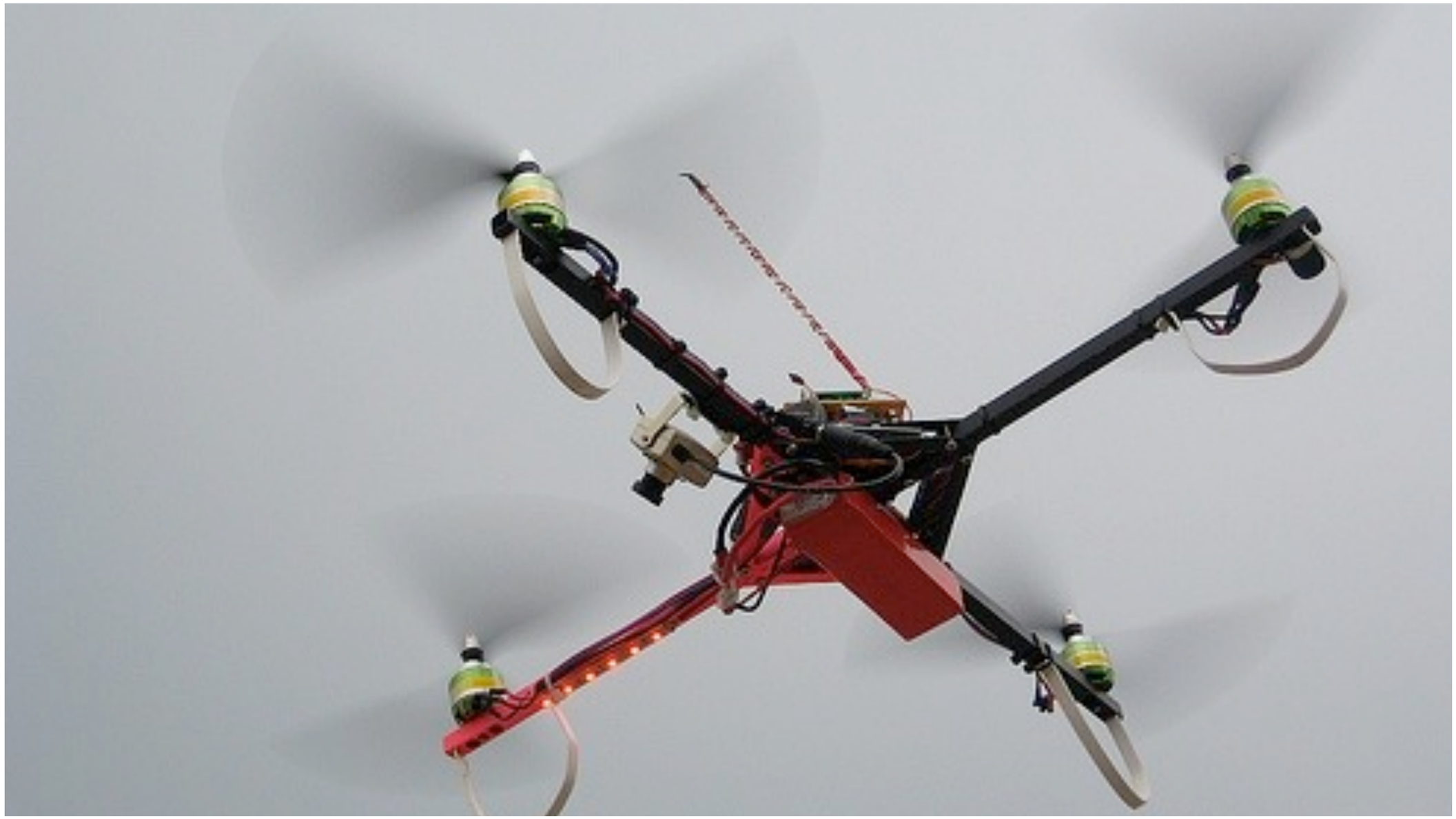
Infrastructure Automation

- Provision servers
- Configure servers
- Configure clustered services
- Deploy software
- Manage software/servers/services

History

http://www.flickr.com/photos/mamalovesyou

Started in 2010 by Hugo Duncan to provide first class support for building and configuring clusters in the cloud



Fun with Quadcopters!

Author: http://www.flickr.com/photos/samchurchill/

complex, but known

Author: http://www.flickr.com/photos/sfslim/

Server Setup

- Create servers in the Cloud: jclouds, fog...
- Install packages: apt, yum...
- Download files: wget, curl...
- Edit files: vi, sed...
- Manage OS: unix tools, PowerShell

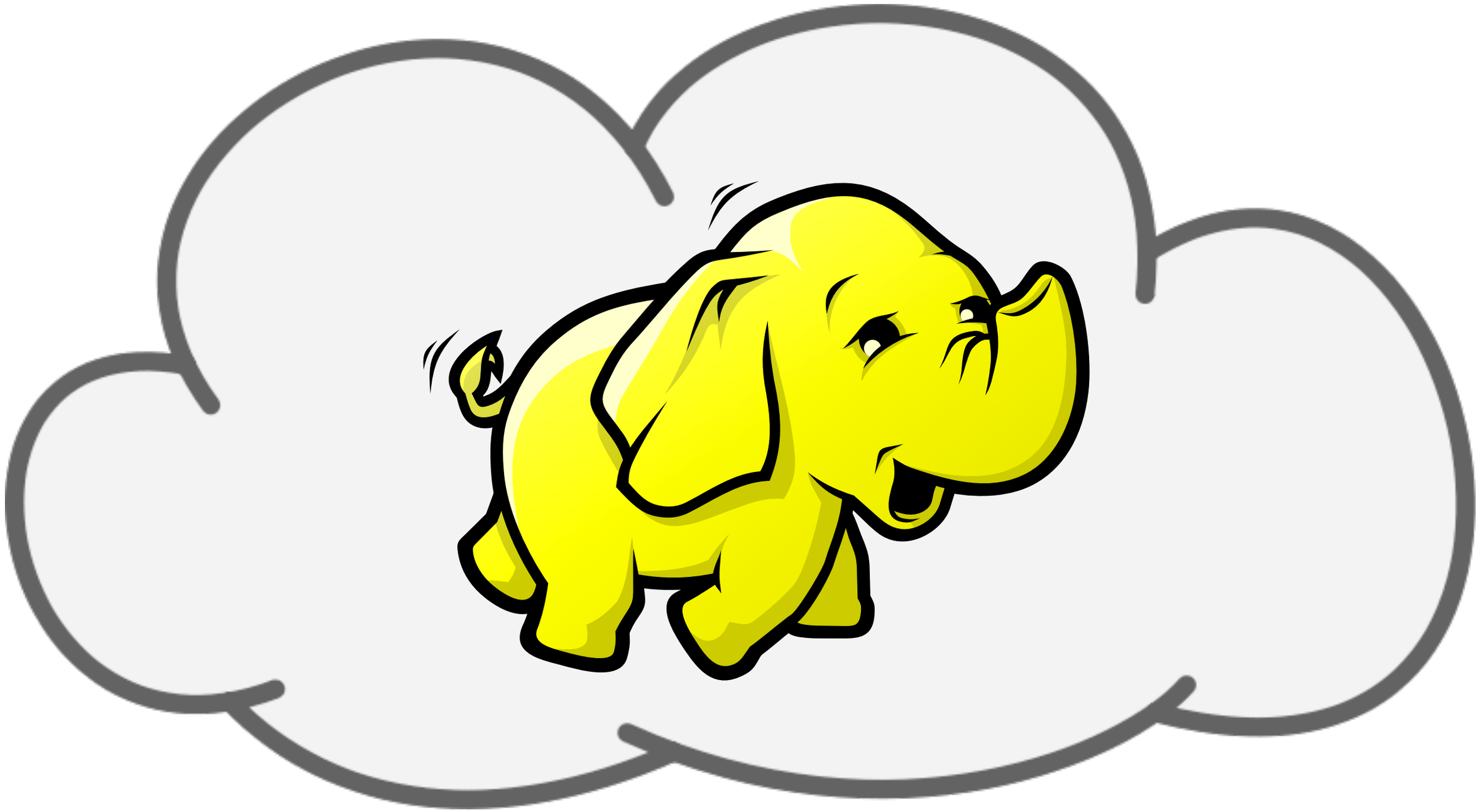
Author: http://www.flickr.com/photos/sfslim/

what if we abstract away the complexity?

like this ^^^ ...but with servers

Pallet is

- a library
- extensible
- works with every cloud (via jclouds) and more...
- build your abstractions
- provisioning, configuration, orchestration



building hadoop clusters
... in the cloud

NameNode

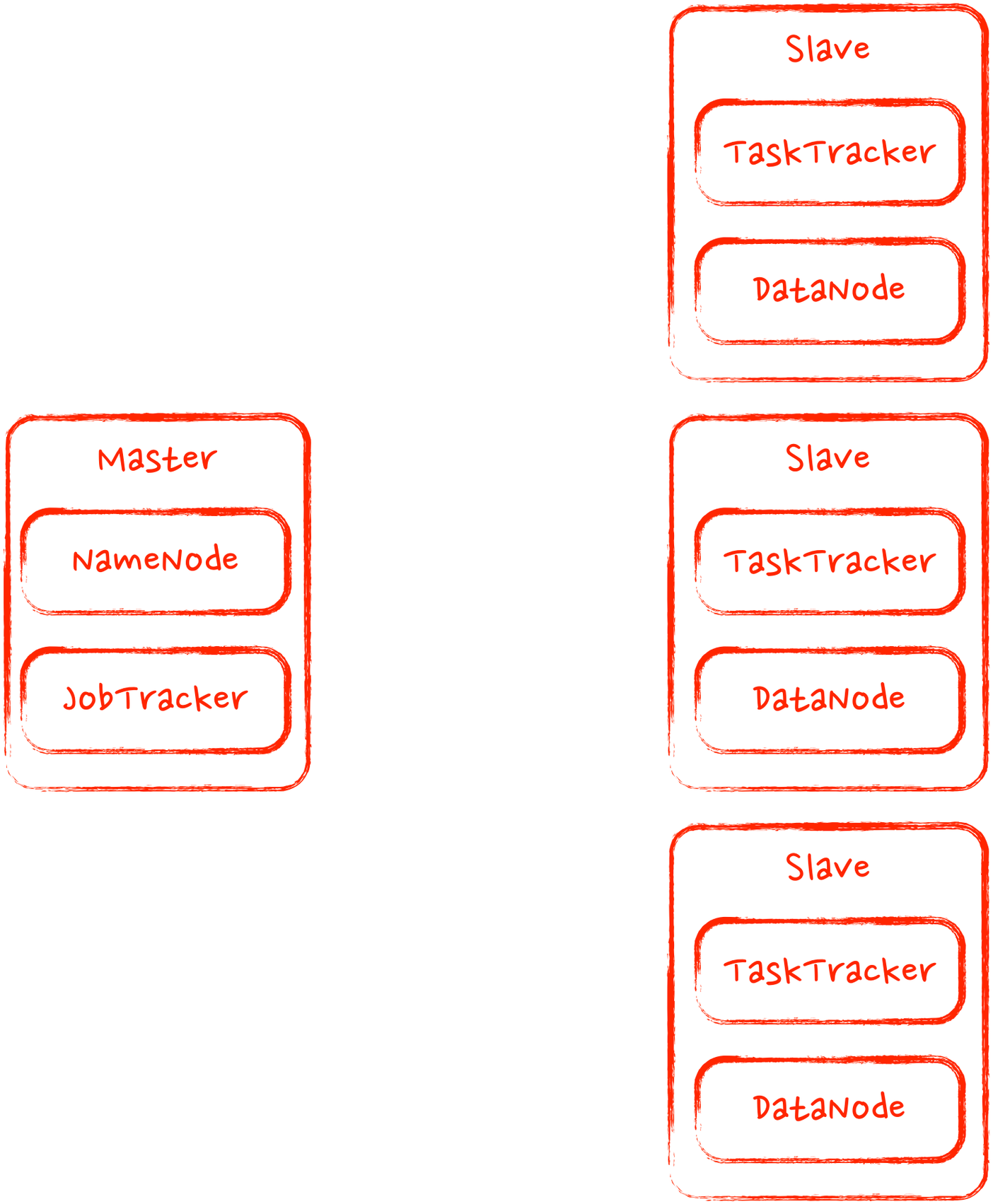
TaskTracker

JobTracker

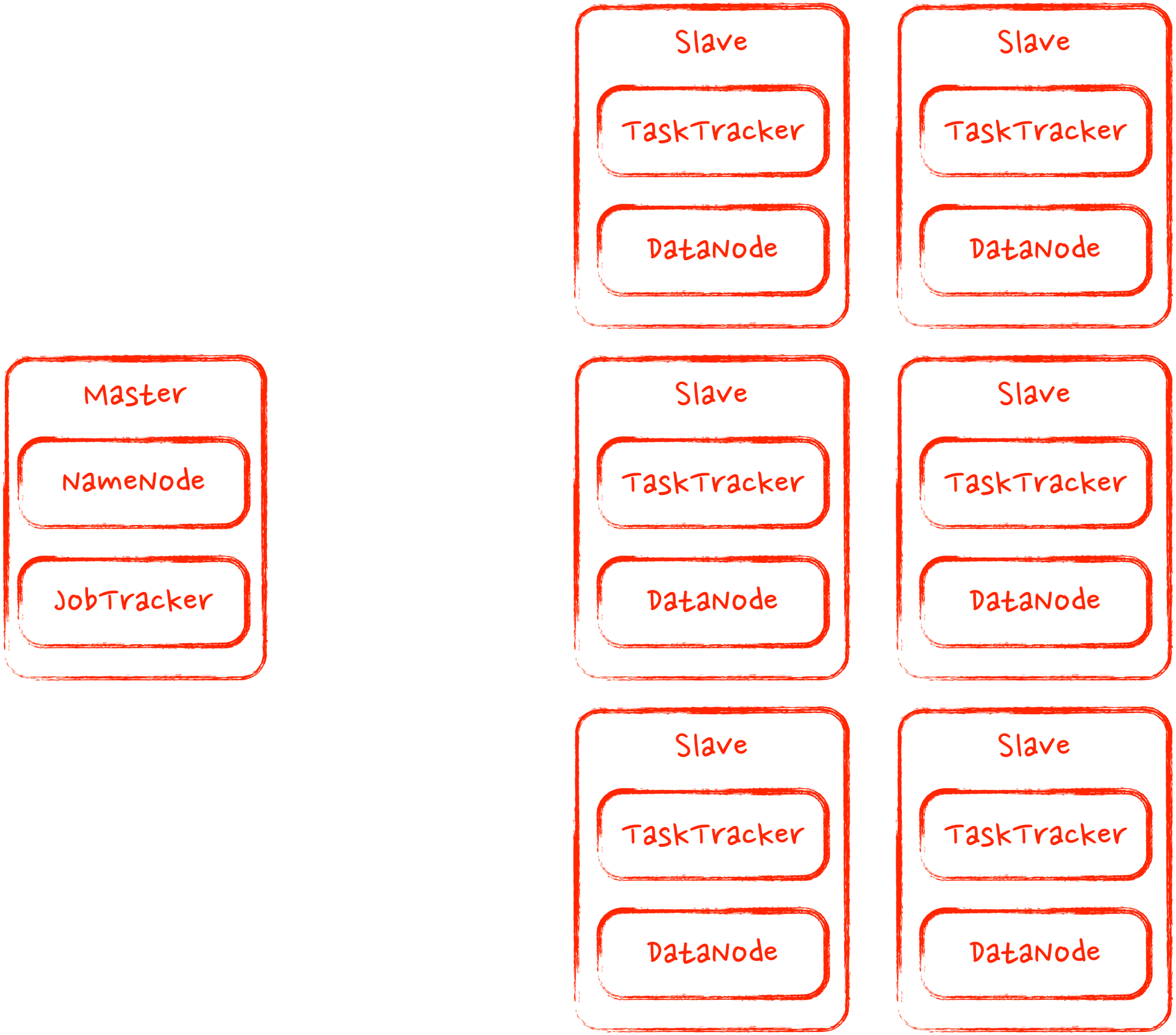
DataNode

caution: Major oversimplification in progress!

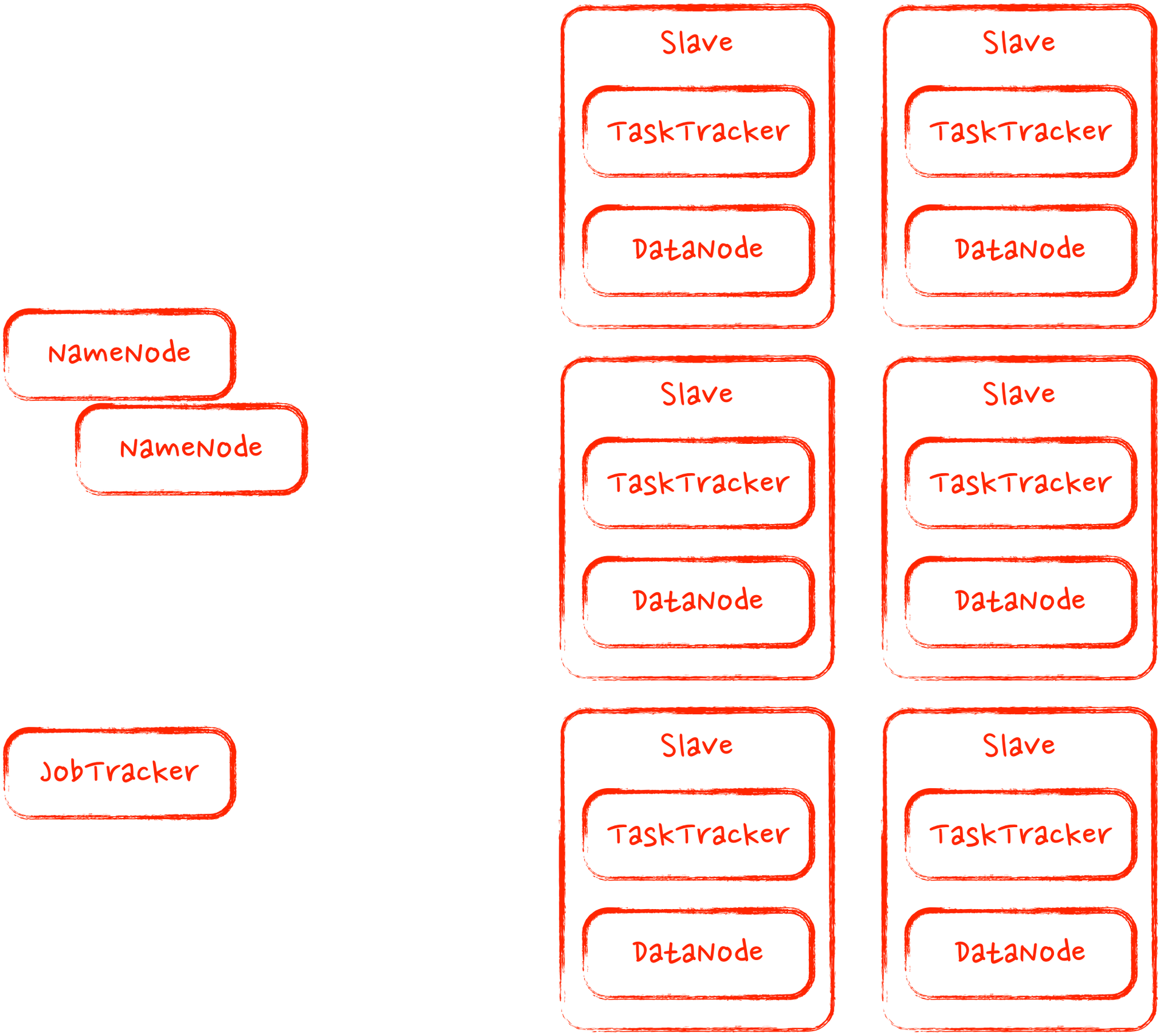
caution: Major oversimplification in progress!



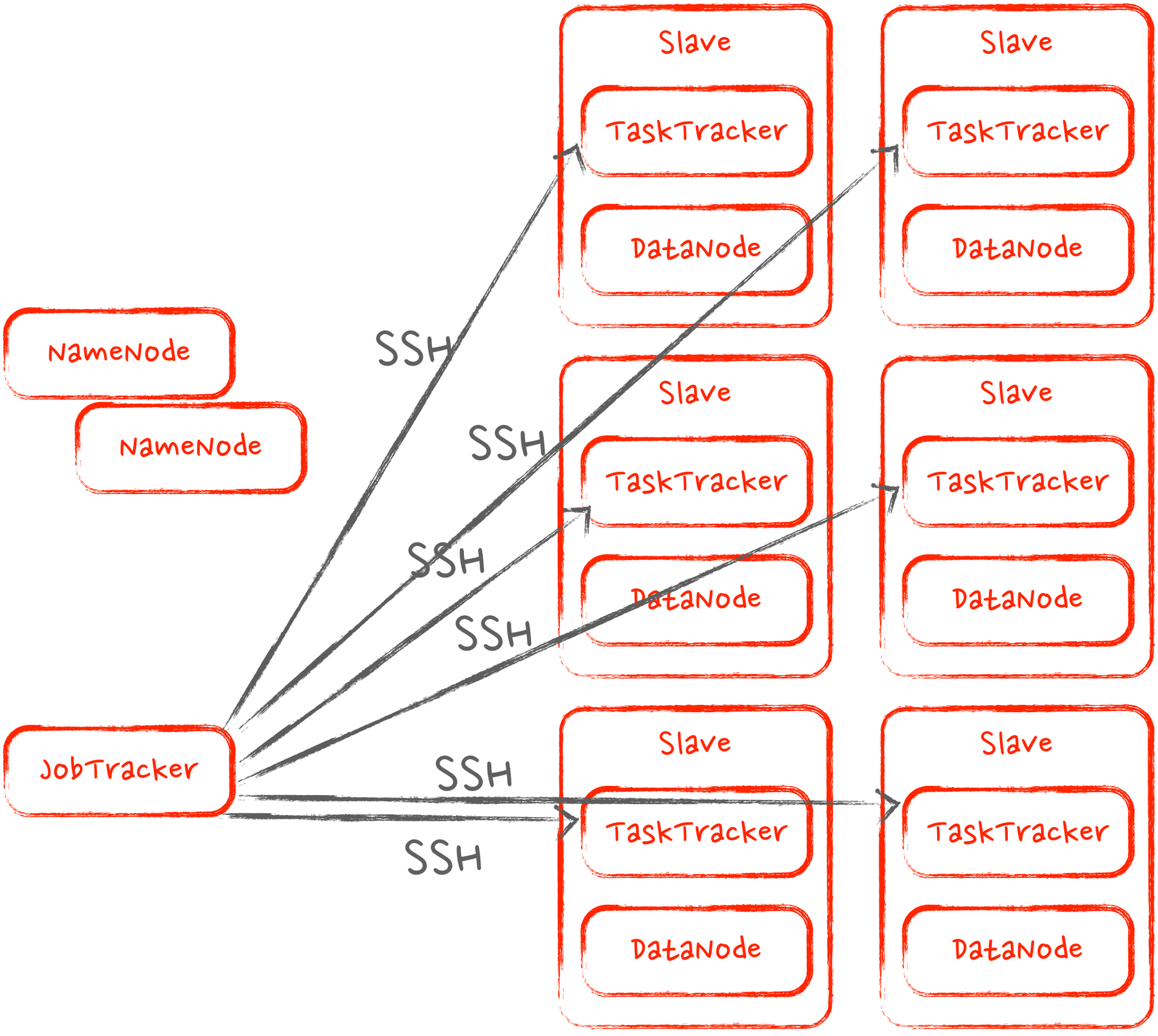
caution: Major oversimplification in progress!



caution: Major oversimplification in progress!



caution: Major oversimplification in progress!



caution: Major oversimplification in progress!

let’s see how

node = class of compute server to be instantiated in the cloud

```
(def small-node
   (node-spec
     :image {:os-family :ubuntu
             :os-version-matches "10.10"}
    :hardware {:min-cores 2 :min-ram 512}
    :network {:inbound-ports [22 80]}))
```

server spec = abstraction over a set of actions
to configure a server

```
(def hadoop
  (server-spec
   :phases
   {:configure
    (phase-fn
     (java/java :jdk)
     (hadoop/install :cloudera))}))
```

```
(def task-tracker
  (server-spec
    :phases
    {:start
     (phase-fn (hadoop/task-tracker))}))

(def job-tracker
  (server-spec
    :phases
    {:start
     (phase-fn (hadoop/job-tracker))}))
```

```
(def task-trackers
  (group-spec "task-tracker"
              :extends [hadoop task-tracker]
              :node-spec big-node))
```

```
(def job-trackers
  (group-spec "job-tracker"
              :extends [hadoop job-tracker]
              :node-spec small-node))
```

Action!

```
(converge {task-trackers 10
           job-trackers 1}
          :phase [:configure :start]
          :compute aws-ec2)
```

... or even...

```
(converge {task-trackers 5
           job-trackers 1}
          :phase [:configure :start]
          :compute virtualbox)
```

crate function = schedules defines actions to be executed on the target nodes

```
(def-crate-fn authorize-key
  "Authorize a public key on the specified user."
  [user public-key-string
    & {:keys [authorize-for-user]}]
  ...
  (directory dir :owner target-user :mode "755")
  (file auth-file :owner target-user :mode "644")
  (exec-checked-script
   (format "authorize-key on user %s" user)
   (echo (quoted ~public-key-string)
         ">>" ~auth-file)))
```

stevedore = A DSL for shell scripts.

```
(script
 (doseq [x ["a" "b" "c"]]
   (println @x)))
```

```
for x in a b c; do
  echo ${x}
done
```

Pallet will turn Stevedore into valid shell scripts (e.g. bash) for the target OS, distribution and version.

First Class Relationships

```
(defn- get-node-ids-for-group
  "Get the id of the nodes in a group node"
  [request group]
  (let [nodes (session/nodes-in-group request group)]
    (map compute/id nodes)))
```

```
(defn- get-keys-for-group
  "Returns the ssh key for a user in a group"
  [request group user]
  (for [node (get-node-ids-for-group request group)]
    (parameter/get-for request [:host (keyword node)
                                :user (keyword user)
                                :id_rsa])))
```

how does it look?

```
(defn make-hadoop-cluster
  [slave-count ram-size-in-mb]
  (cluster-spec
    :private
    {:jobtracker (node-group [:jobtracker :namenode])
     :slaves     (slave-group slave-count)}
     :base-machine-spec {:os-family :ubuntu
                         :os-version-matches "10.10"
                         :os-64-bit true
                         :min-ram ram-size-in-mb}
   :base-props {:hdfs-site
                {:dfs.data.dir "/mnt/dfs/data"
                 :dfs.name.dir "/mnt/dfs/name"}
                :mapred-site {...}}))
```

```
(create-cluster
     (make-hadoop-cluster 30 (* 8 1024))
     aws-ec2)
```

https://github.com/pallet/pallet-hadoop-example

One Source to

- develop your app on a local vm
- setup a integration test server
- create a staging environment
- create a production
- N instances
- anywhere

...There is Much More...

- cluster-specs
- roles
- all cloud providers (via jclouds)
- external resources
- hybrid clouds

FIN

palletops.com