# k8s : h4p

*From docker to kubernetes:*
*Running Apache Hadoop in a cloud native way*

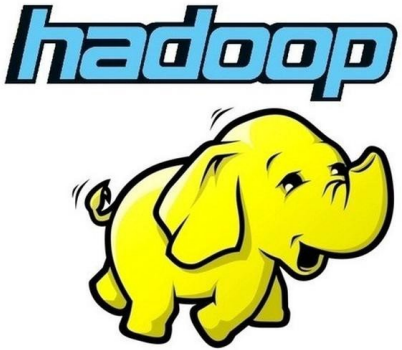
Marton Elek

RATIS

RATIS

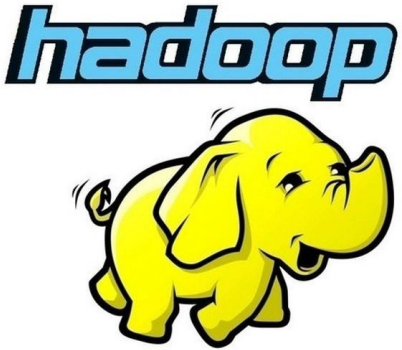
# RATIS


RAFT library for Java

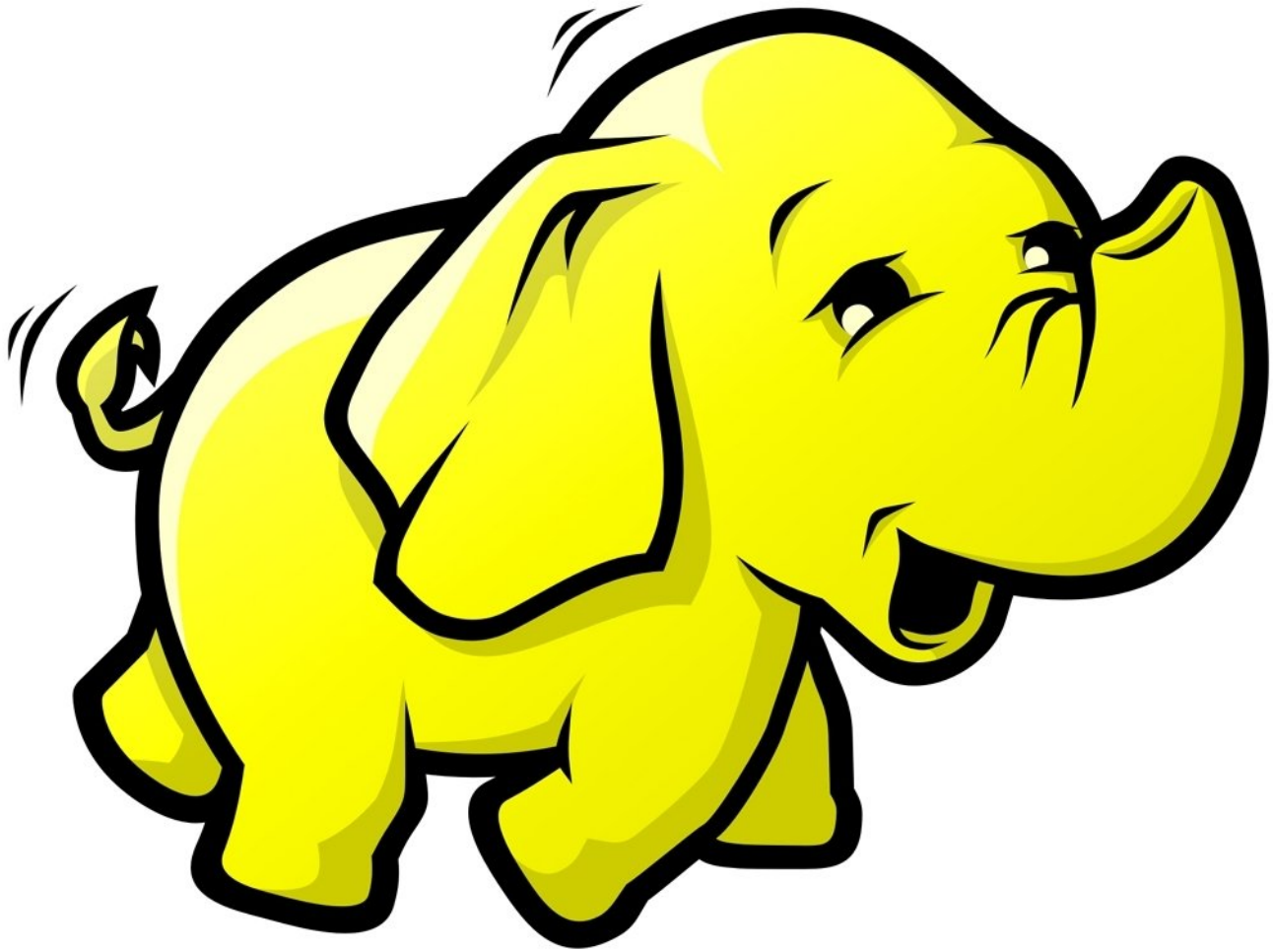


# RATIS


RAFT library for Java




https://flokkr.github.io




©2007 Cizauskas

# Energy

Manufacturer
Model

Fridge-Freezer

**More efficient**

A
B
C
D
E
F
G

**Less efficient**

A

Energy consumption kWh/year
(Based on standard test results for 24h)

Actual consumption will depend on how the appliance is used and where it is located

325

| | |
|---|---|
| Fresh food volume l | 190 |
| Frozen food volume l | 126 |
| | ✱✱✱✱ |

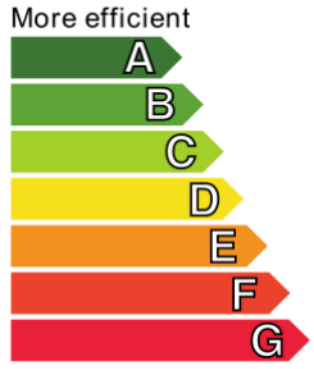
**Noise**
(dB(A) re 1 pW)

Further information is contained in product brochures

Norm EN 153 May 1990
Refrigerator Label Directive 94/2/EC

# Docker

Local compose file



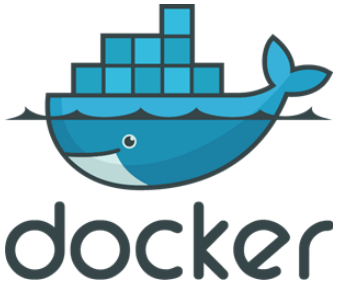


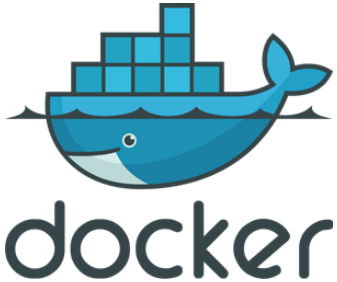

| | |
|---|---|
| **Configuration management** | |
| Source | |
| Preprocessing | |
| On change | |
| **Provisioning, Scheduling** | |
| Multihost support | |
| Scheduling | |
| Cluster definition | |
| Scaling | |
| Multi tenancy | |
| Failover | |
| **Network** | |
| Intraservice network | |
| DNS | |
| Service discovery | |
| Data locality | |
| Availability of the ports | |

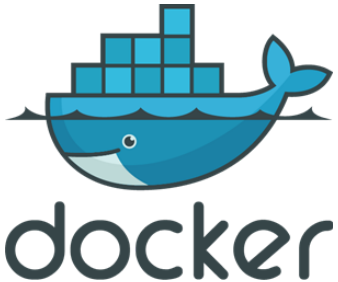


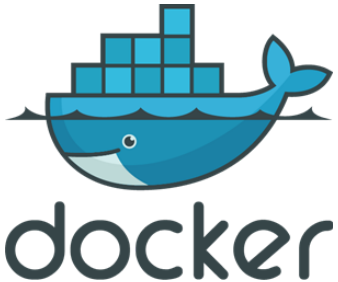
| | |
|---|---|
| **Configuration management** | |
| Source | |
| Preprocessing | |
| On change | |
| **Provisioning, Scheduling** | |
| Multihost support | |
| Scheduling | |
| Cluster definition | |
| Scaling | |
| Multi tenancy | |
| Failover | |
| **Network** | |
| Intraservice network | |
| DNS | |
| Service discovery | |
| Data locality | |
| Availability of the ports | |

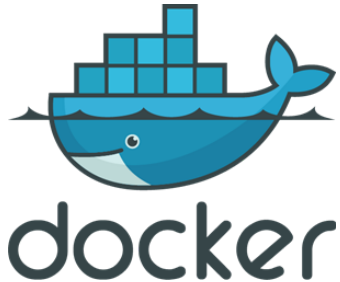
docker

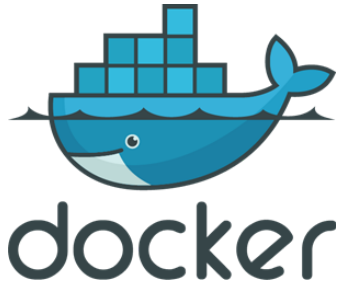

docker


docker


docker


docker


docker


docker

# Dockerfile

```
FROM frolvlad/alpine-oraclejdk8
ADD hadoop-3.2.0.tar.gz /opt
WORKDIR /opt/hadoop
```


| | |
|---|---|
| **Configuration management** | |
| Source | |
| Preprocessing | |
| On change | |
| **Provisioning, Scheduling** | |
| Multihost support | |
| Scheduling | |
| Cluster definition | |
| Scaling | |
| Multi tenancy | |
| Failover | |
| **Network** | |
| Intraservice network | |
| DNS | |
| Service discovery | |
| Data locality | |
| Availability of the ports | |

```xml
<configuration>
   <property>
      <name>dfs.namenode.rpc-address</name>
      <value>namenode:9000</value>
   </property>
   <property>
      <name>dfs.datanode.plugins</name>
      <value>org.apache.hadoop.ozone.HddsDatanodeService</value>
   </property>
   <property>
      <name>rpc.metrics.percentiles.intervals</name>
      <value>60,300</value>
   </property>
   <property>
      <name>dfs.namenode.name.dir</name>
      <value>/data/namenode</value>
   </property>
   <property>
      <name>rpc.metrics.quantile.enable</name>
      <value>true</value>
   </property>
</configuration>
```

```yaml
version: "3"
services:
   service1:
      image: apache/imagename
      hostname: namenode
      ports:
         - 9870:9870
      environment:
          CONFIGURATION1: value
          DFS_DIR: /dfs
          THREAD_NUMBER: 1
```

# How to handle configuration?

Create a simple **launcher** script to

- Create config file from environment variables
- Start the application

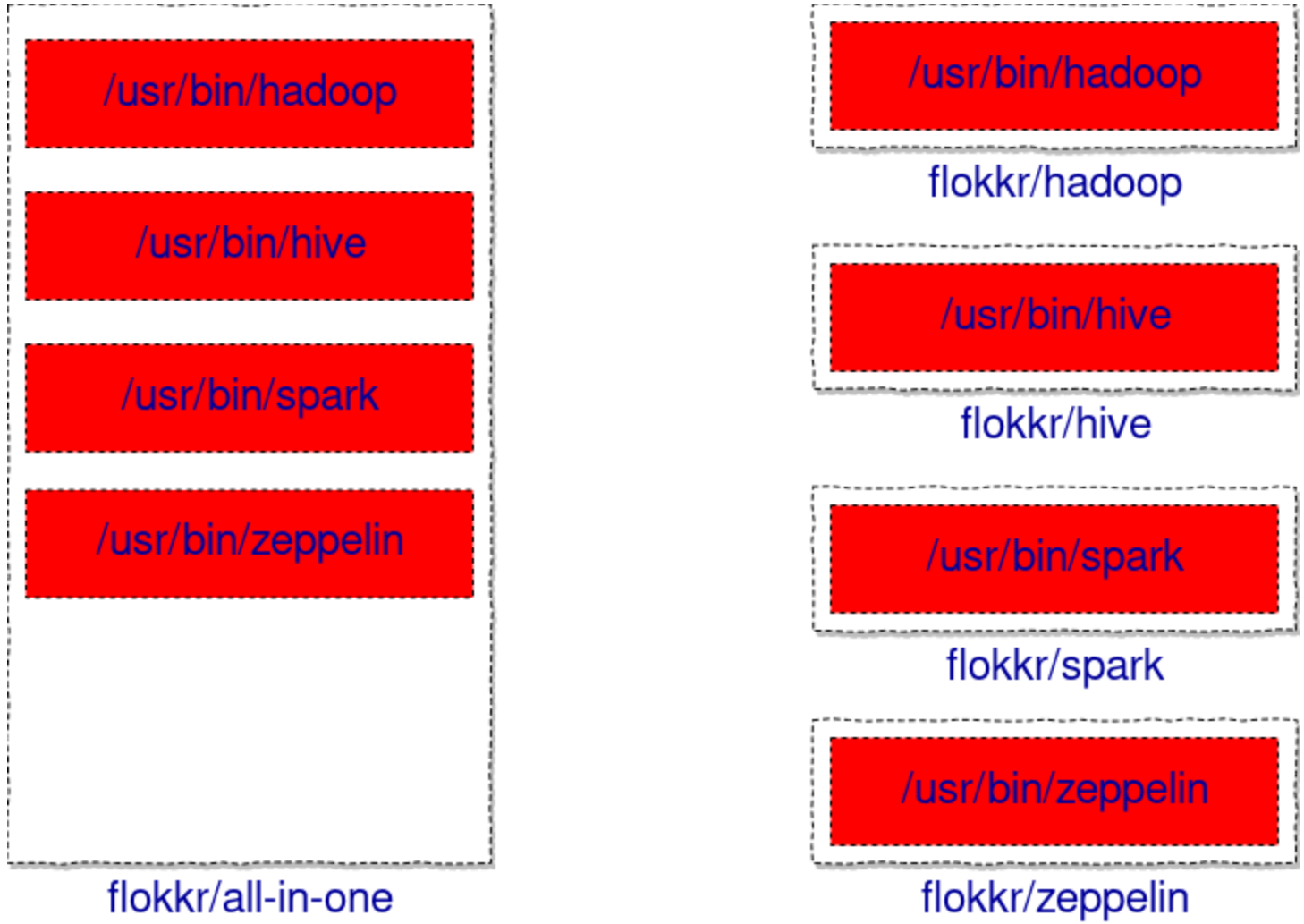
```yaml
version: "3"
services:
   namenode:
      image: flokkr/hadoop
      hostname: namenode
      command: ["hdfs","namenode"]
      ports:
         - 9870:9870
      environment:
         ENSURE_NAMENODE_DIR: "/tmp/hadoop-root/dfs/name"
         CORE-SITE.XML_fs.defaultFS: "hdfs://namenode:9000"
         HDFS-SITE.XML_dfs.namenode.rpc-address: "namenode:9000"
         HDFS-SITE.XML_dfs.replication: "1"
   datanode:
      image: flokkr/hadoop
      command: ["hdfs", "datanode"]
      environment:
         CORE-SITE.XML_fs.defaultFS: "hdfs://namenode:9000"
         HDFS-SITE.XML_dfs.namenode.rpc-address: "namenode:9000"
         HDFS-SITE.XML_dfs.replication: "1"
         LOG4J.PROPERTIES_log4j.rootLogger: "INFO, stdout"
         LOG4J.PROPERTIES_log4j.appender.stdout: "org.apache.log4j.ConsoleAppender"
         LOG4J.PROPERTIES_log4j.appender.stdout.layout: "org.apache.log4j.PatternLayout"
         LOG4J.PROPERTIES_log4j.appender.stdout.layout.ConversionPattern: "%d{yyyy-MM-dd HH:mm:ss} %-5p
```

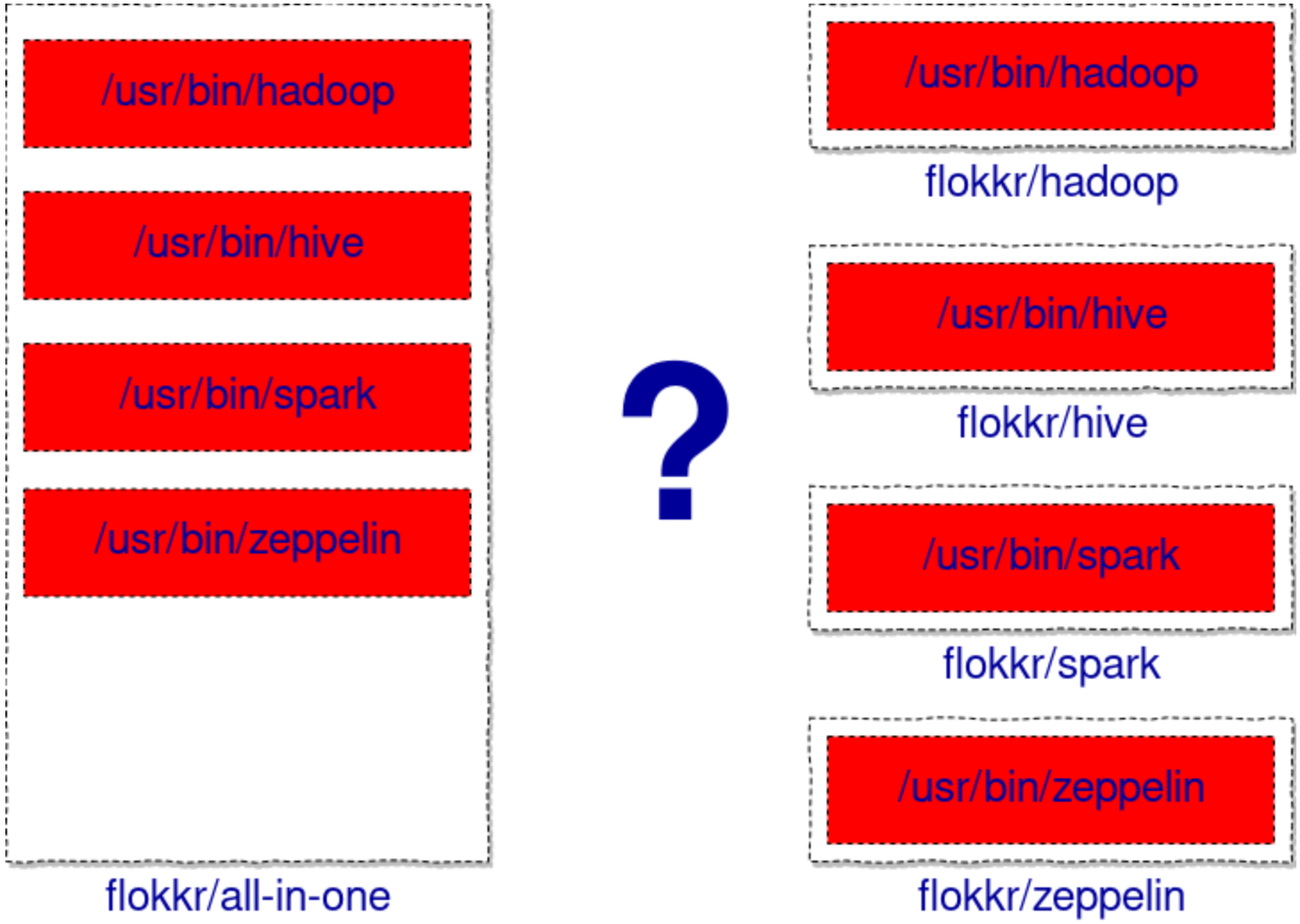

Less efficient

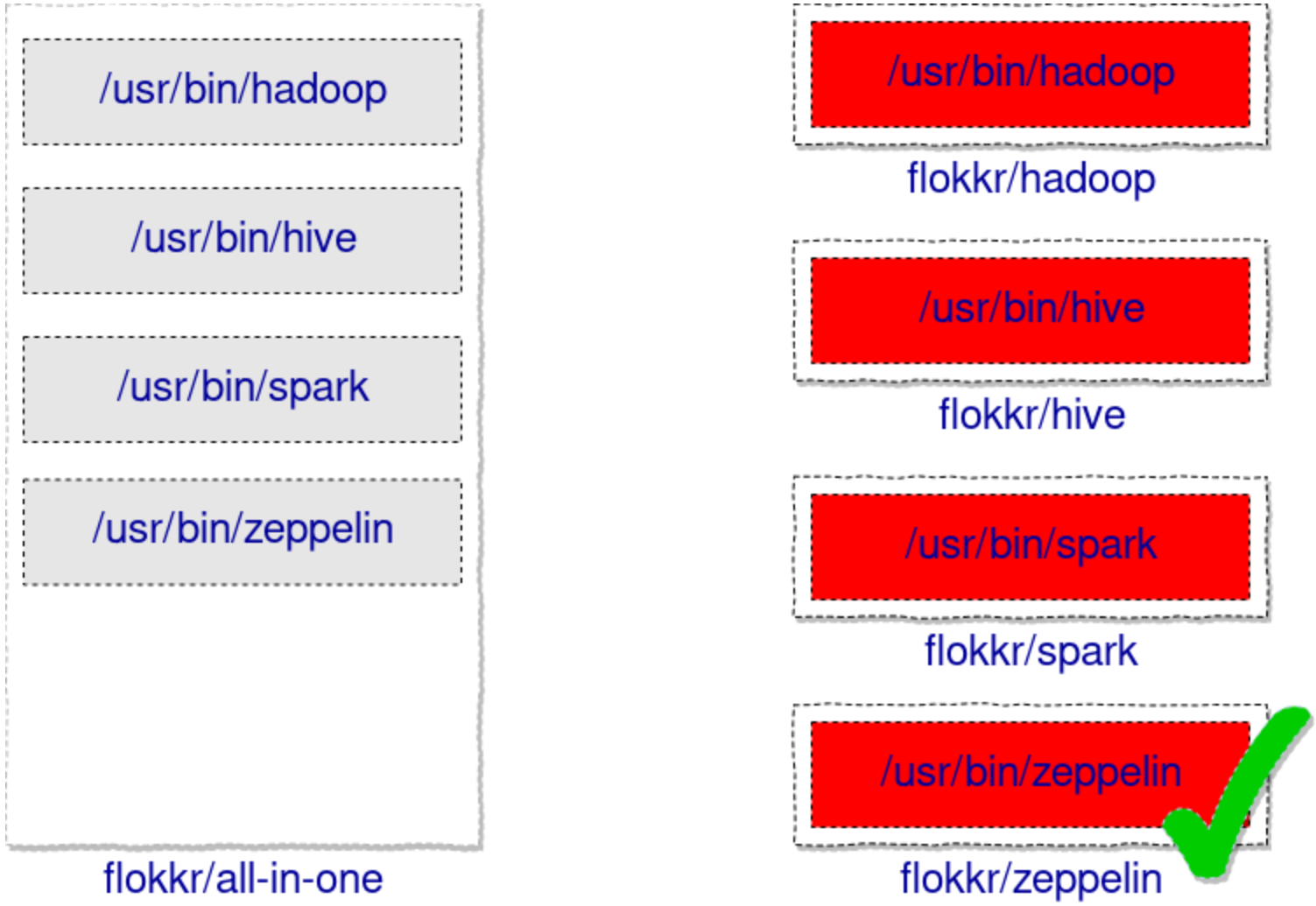
docker

| | |
|---|---|
| **Configuration management** | |
| Source | ENV (script) |
| Preprocessing | n/a |
| On change | n/a |
| **Provisioning, Scheduling** | |
| Multihost support | |
| Scheduling | |
| Cluster definition | |
| Scaling | |
| Multi tenancy | |
| Failover | |
| **Network** | |
| Intraservice network | |
| DNS | |
| Service discovery | |
| Data locality | |


flokkr/all-in-one


/usr/bin/hadoop
/usr/bin/hive
/usr/bin/spark
/usr/bin/zeppelin
flokkr/all-in-one
/usr/bin/hadoop
flokkr/hadoop
/usr/bin/hive
flokkr/hive
/usr/bin/spark
flokkr/spark
/usr/bin/zeppelin
flokkr/zeppelin


/usr/bin/hadoop
/usr/bin/hive
/usr/bin/spark
/usr/bin/zeppelin
flokkr/all-in-one
?
/usr/bin/hadoop
flokkr/hadoop
/usr/bin/hive
flokkr/hive
/usr/bin/spark
flokkr/spark
/usr/bin/zeppelin
flokkr/zeppelin


/usr/bin/hadoop
/usr/bin/hive
/usr/bin/spark
/usr/bin/zeppelin
flokkr/all-in-one
/usr/bin/hadoop
flokkr/hadoop
/usr/bin/hive
flokkr/hive
/usr/bin/spark
flokkr/spark
/usr/bin/zeppelin
flokkr/zeppelin

Container is the **unit** of packaging.

**Launcher** script:
the source of the power in containers

# Dockerfile

```
FROM frolvlad/alpine-oraclejdk8
ADD hadoop-3.2.0.tar.gz /opt
WORKDIR /opt/hadoop
CMD ["/opt/launcher.sh"]
```

# Dockerfile

```
FROM frolvlad/alpine-oraclejdk8
ADD hadoop-3.2.0.tar.gz /opt
WORKDIR /opt/hadoop
CMD ["/opt/launcher.sh"]
```

Final command executed by docker:

```
launcher.sh hdfs namenode
```

```bash
#!/usr/bin/env bash

if [ -n "$SLEEP_SECONDS" ]; then
   echo "Sleeping for $SLEEP_SECONDS seconds"
   sleep $SLEEP_SECONDS
fi


if [ -n "$ENSURE_NAMENODE_DIR" ]; then
   if [ ! -d "$ENSURE_NAMENODE_DIR" ]; then
      /opt/hadoop/bin/hdfs namenode -format -force $CLUSTERID_OPTS
        fi
fi

"$@"
```

# Launcher script

# Launcher script

- Create config files from ENV

# Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)

# Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component

# Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)

# Launcher script

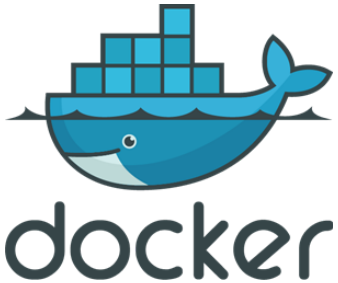
- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)
- Retrieve kerberos/SSL secrets

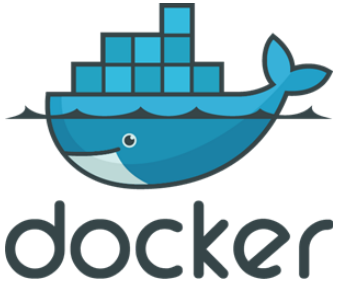
# Launcher script

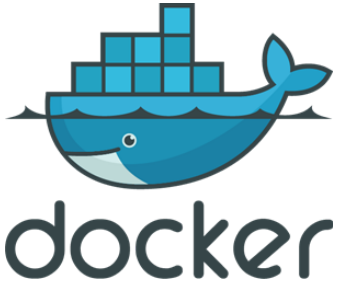
- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)
- Retrieve kerberos/SSL secrets
- Enable prometheus monitoring (Java agent)

# Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)
- Retrieve kerberos/SSL secrets
- Enable prometheus monitoring (Java agent)
- Show network traffic (Instrumentation with Java agent)


docker


docker


docker

# Hashicorp stack

"do it yourself"


Service Discovery and Configuration Made Easy

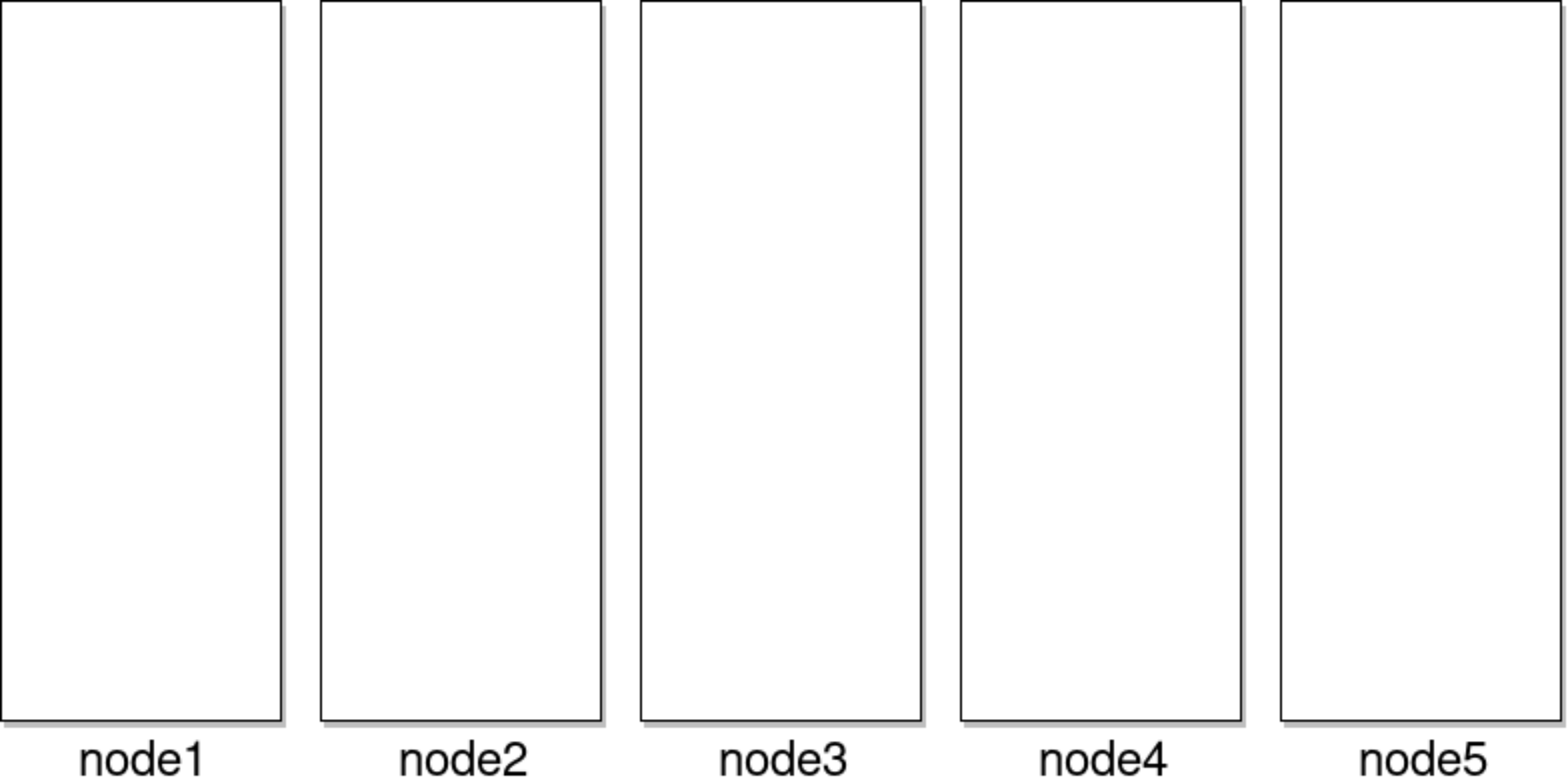

Service Discovery and Configuration Made Easy

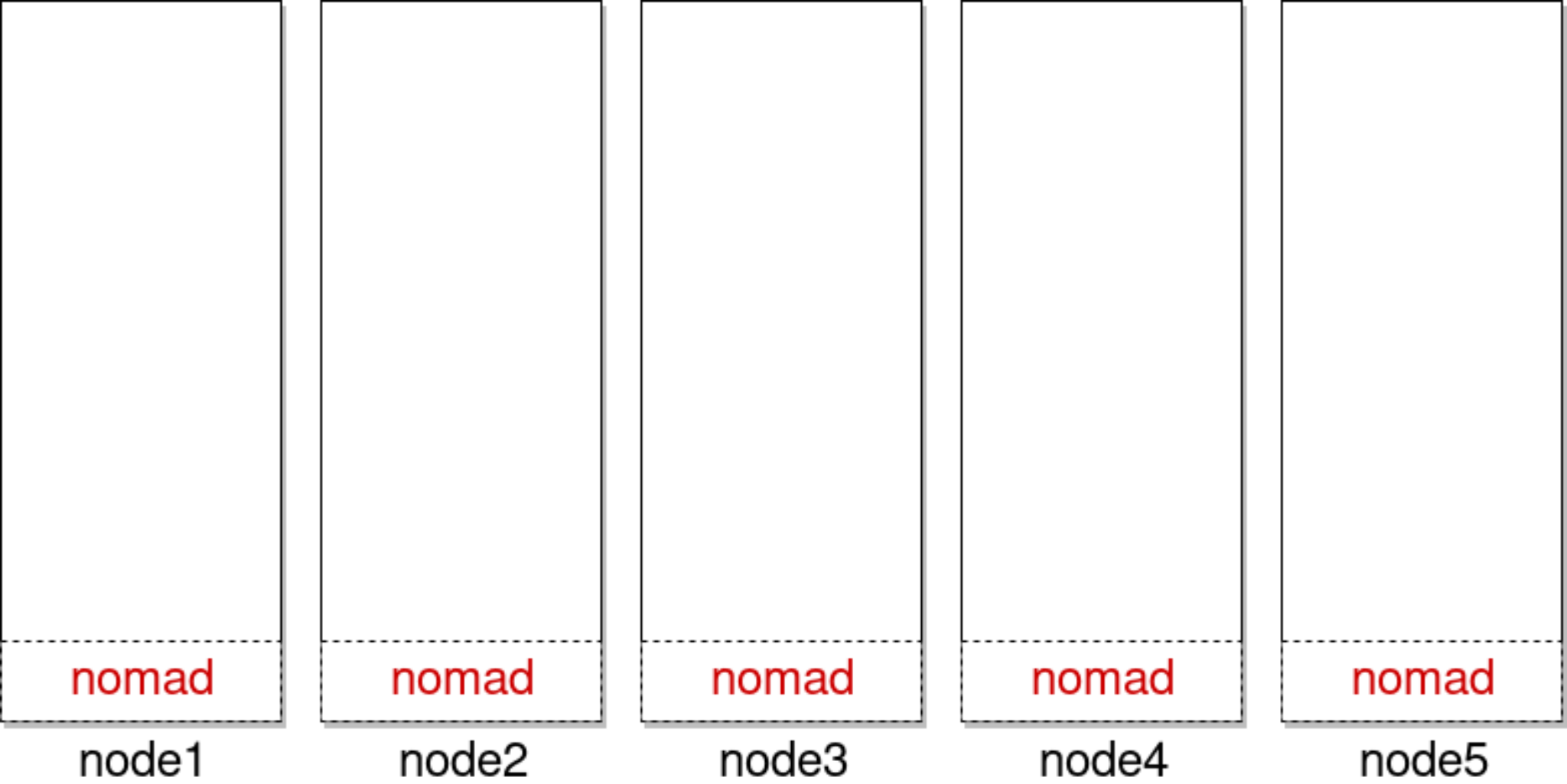

A Tool for Managing Secrets

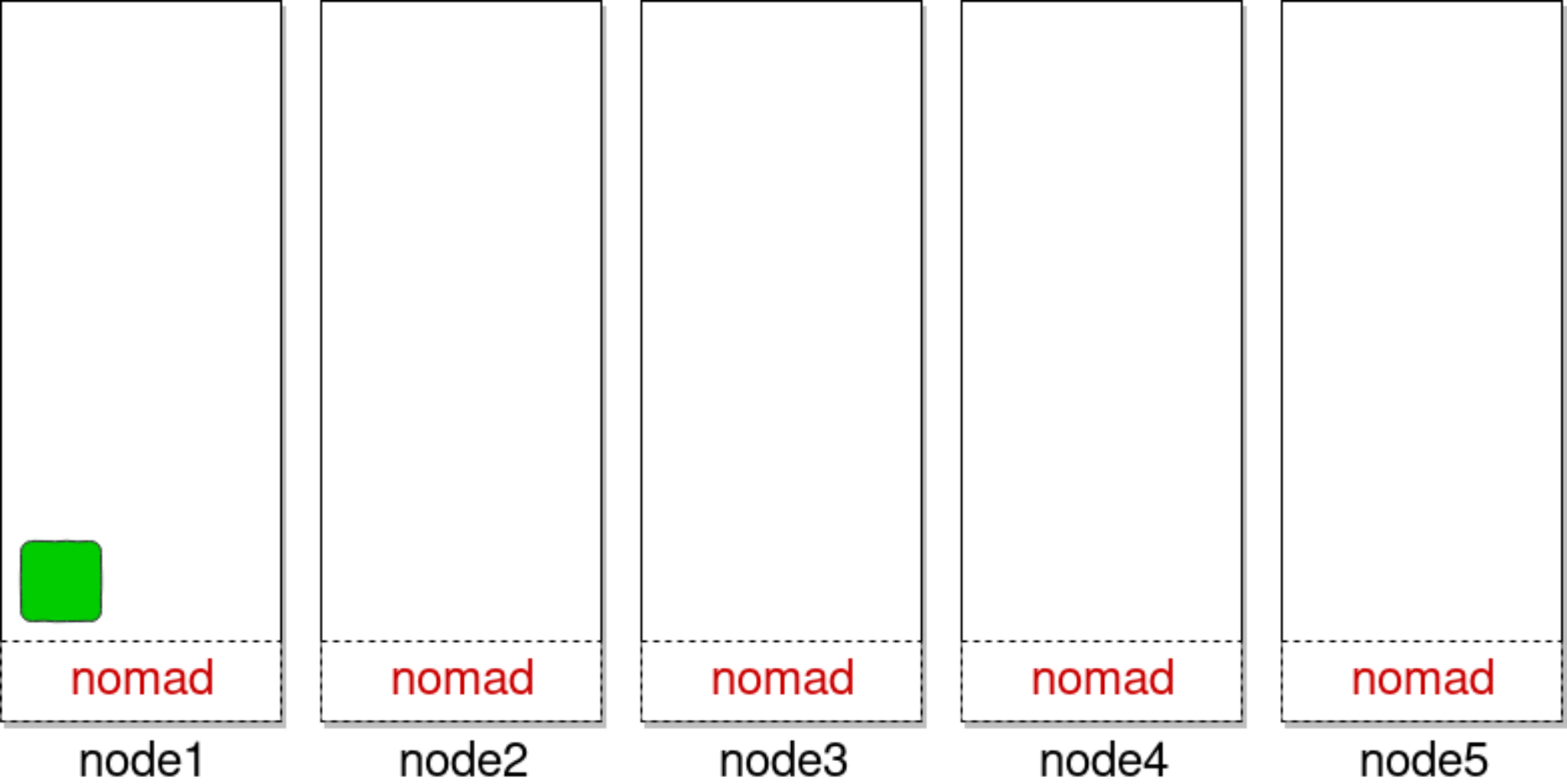

Service Discovery and Configuration Made Easy

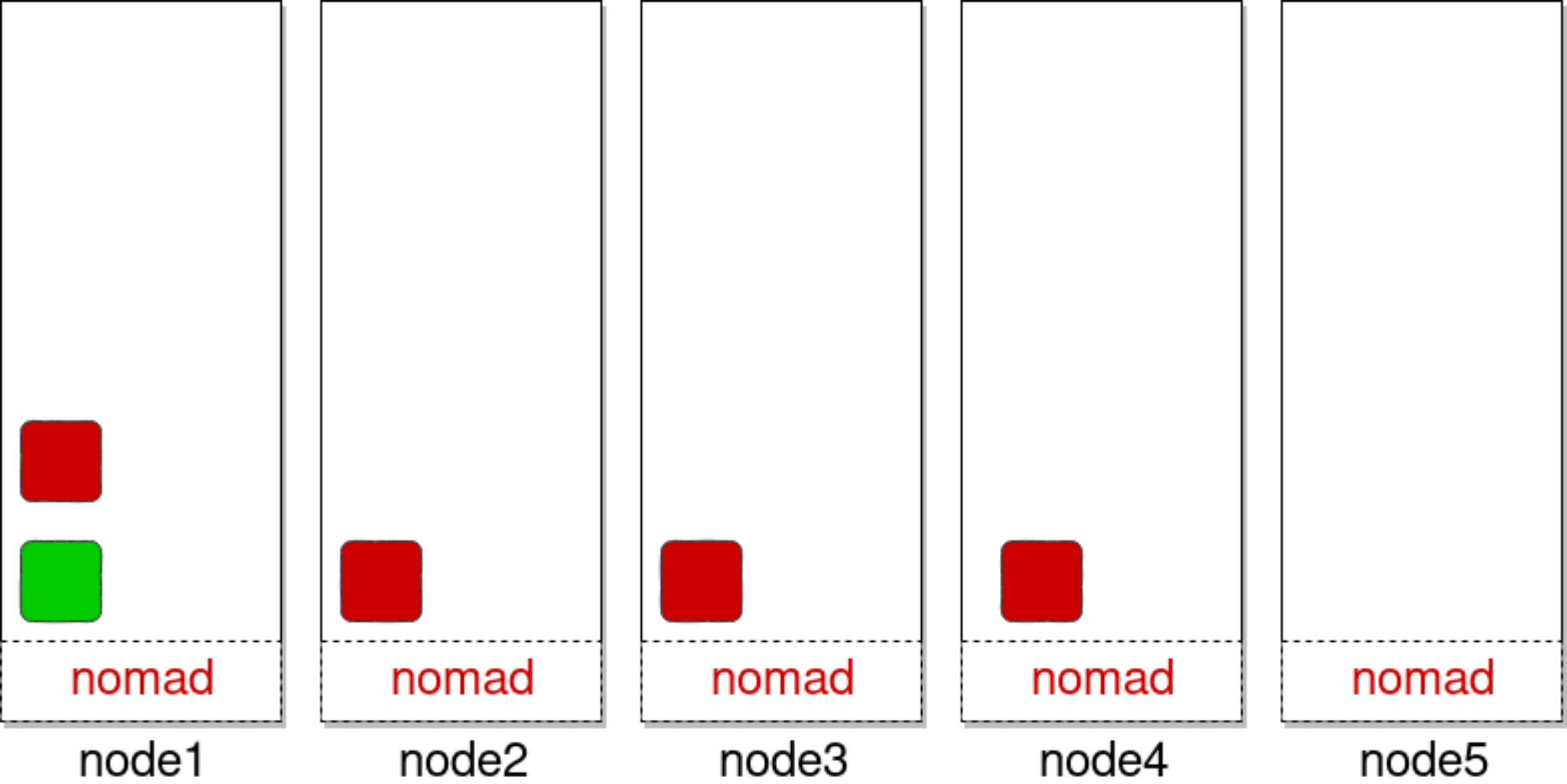

A Tool for Managing Secrets

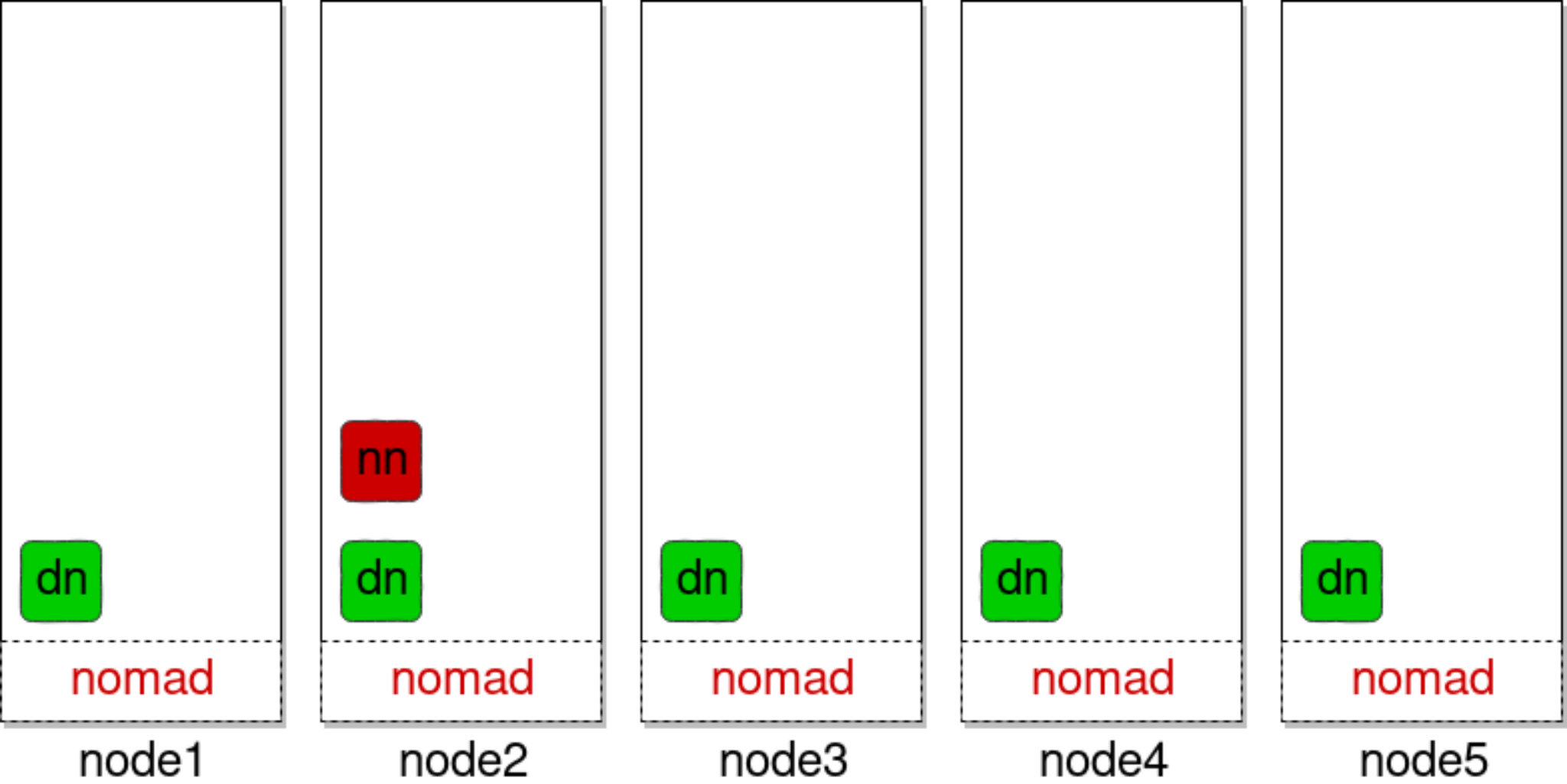

Easily Deploy Applications at Any Scale

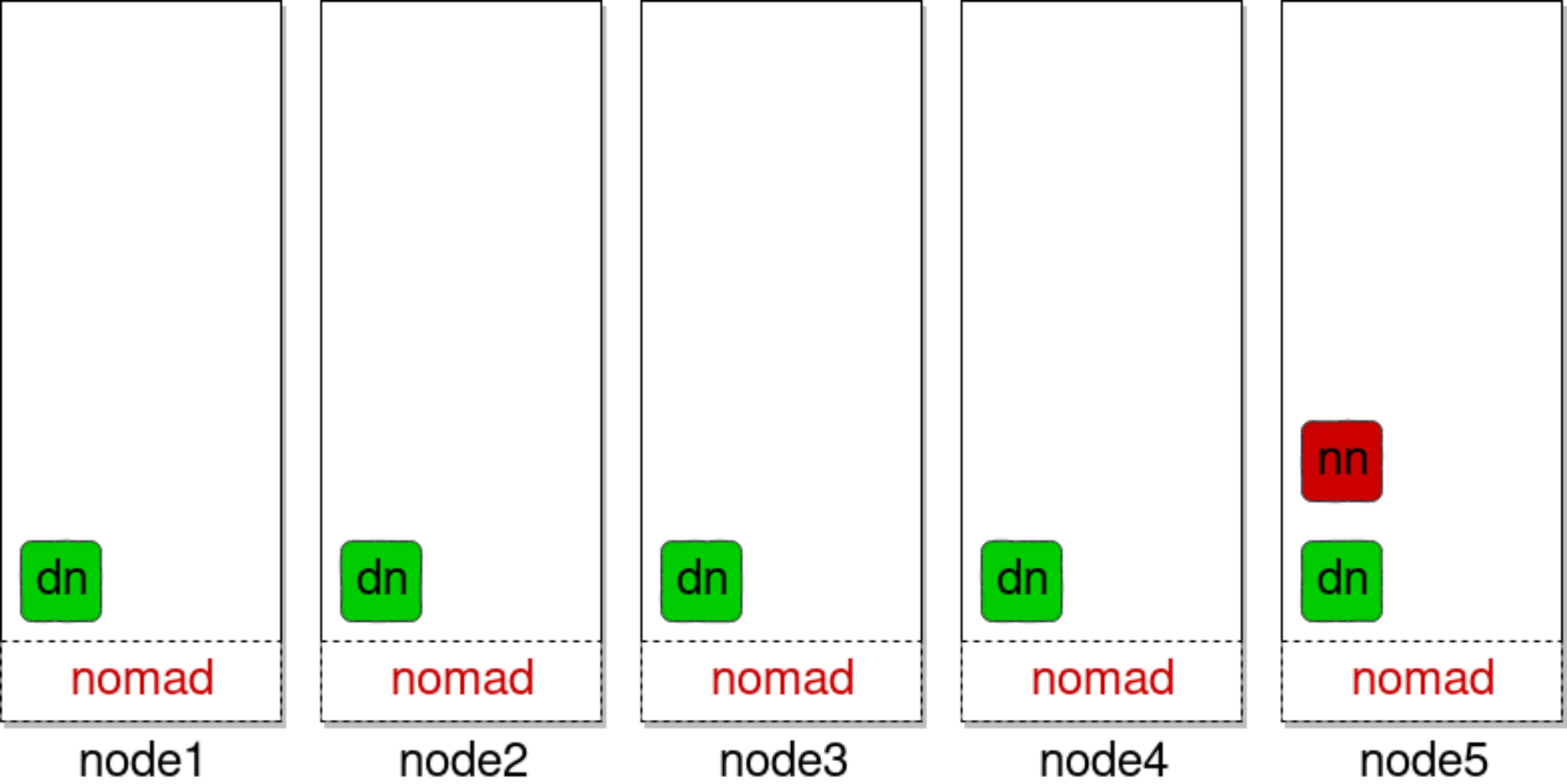

node1
node2
node3
node4
node5

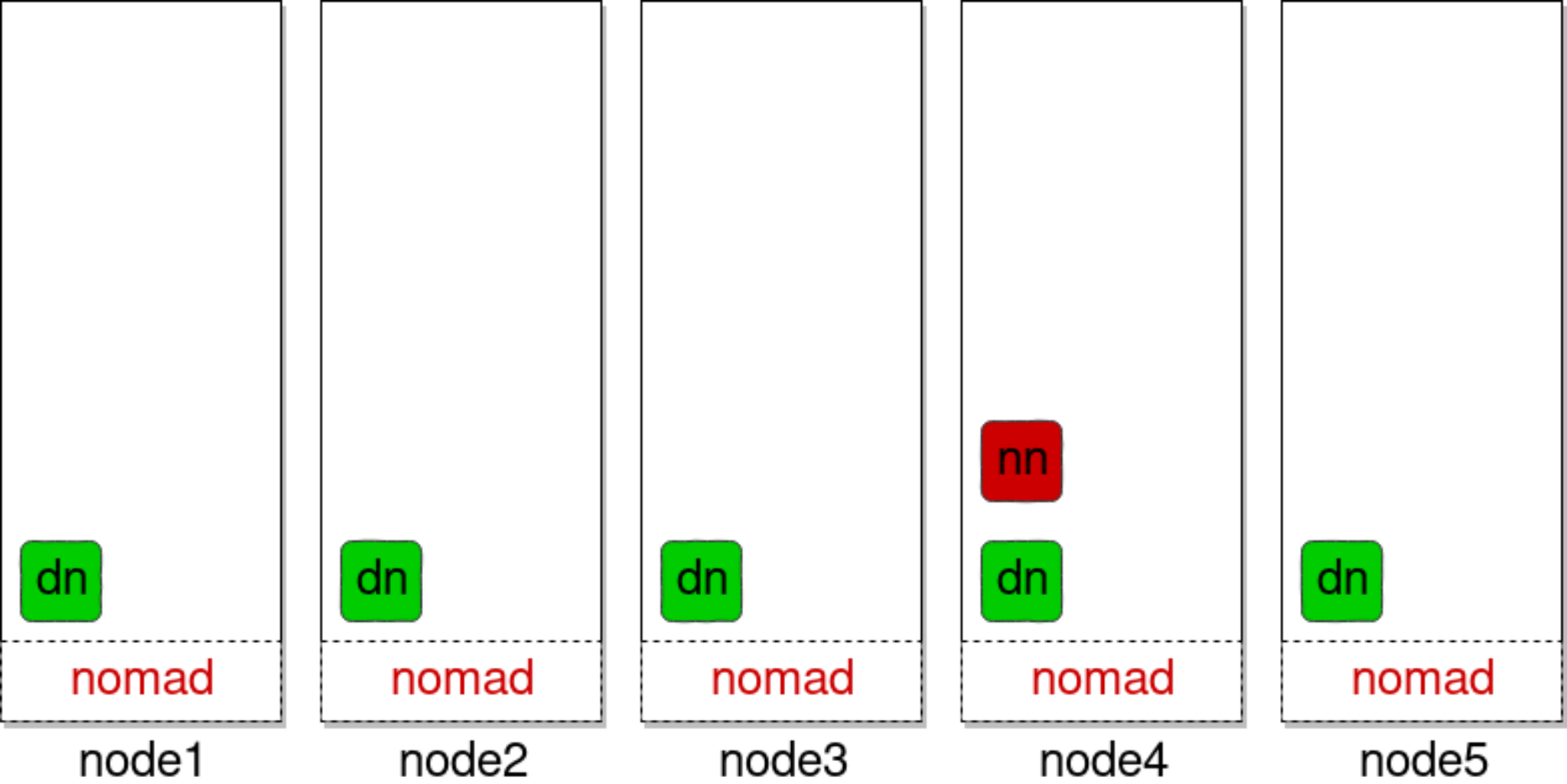

nomad
nomad
nomad
nomad
nomad
node1
node2
node3
node4
node5


nomad
nomad
nomad
nomad
nomad
node1
node2
node3
node4
node5

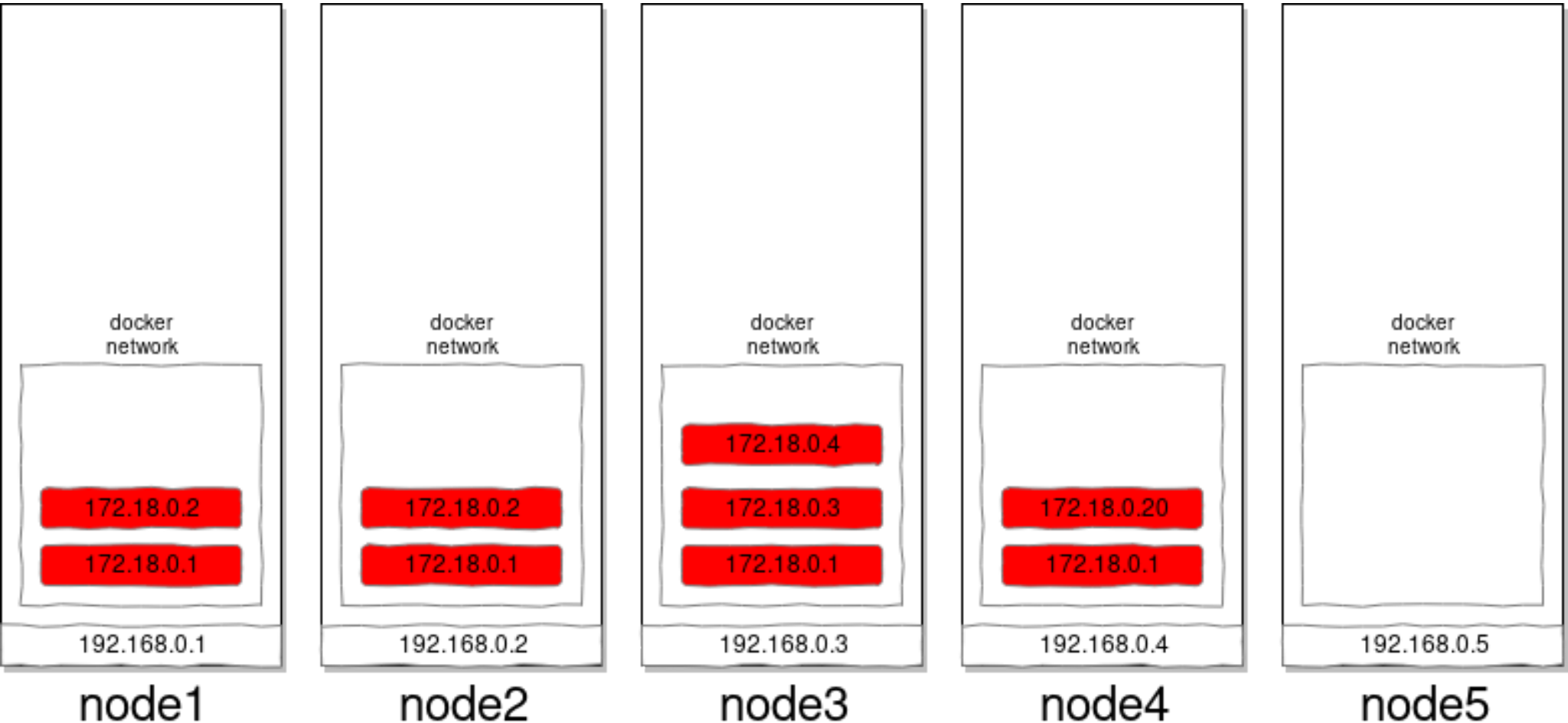

nomad
nomad
nomad
nomad
nomad
node1
node2
node3
node4
node5


nn
dn
dn
dn
dn
dn
nomad
nomad
nomad
nomad
nomad
node1
node2
node3
node4
node5

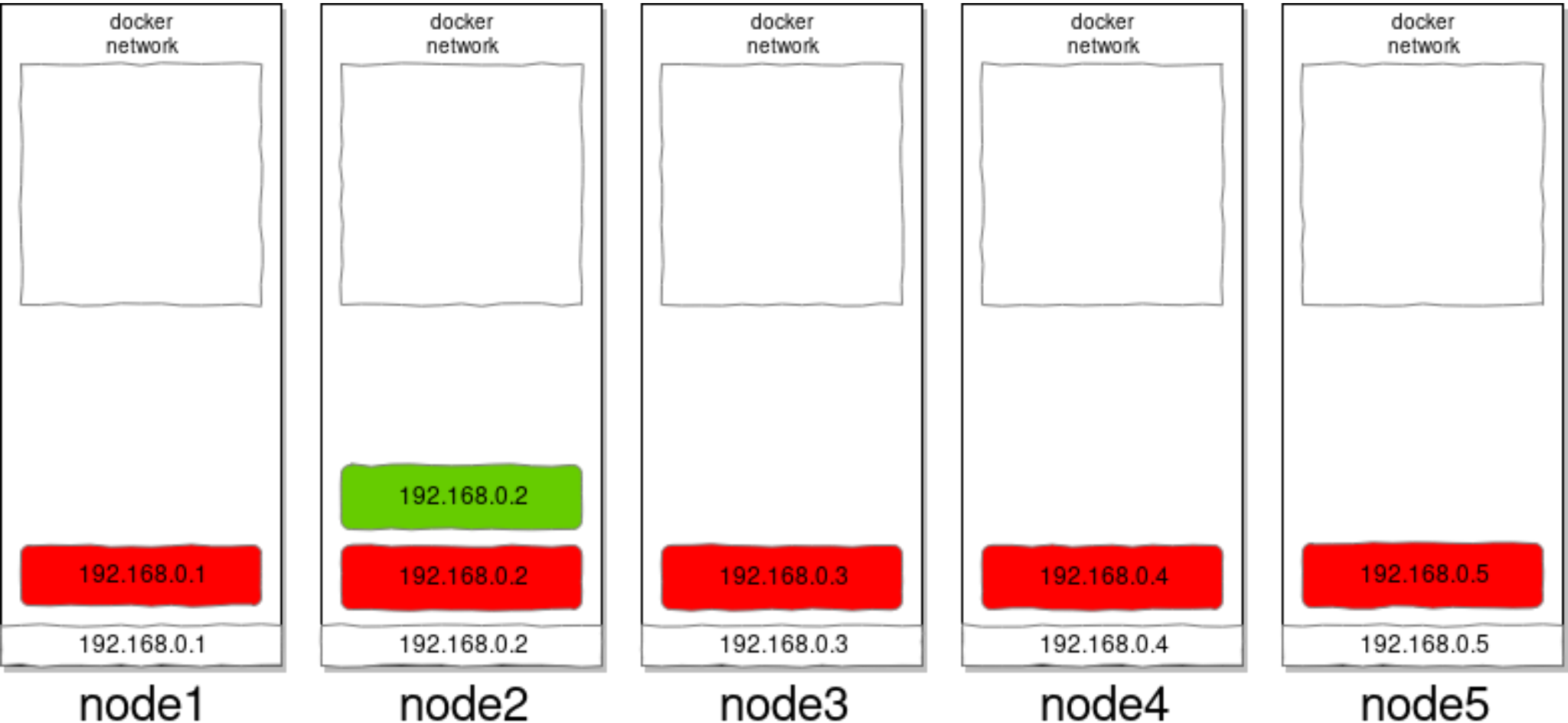

dn
nomad
node1
dn
nomad
node2
dn
nomad
node3
dn
nomad
node4
nn
dn
nomad
node5


dn
nomad
node1
dn
nomad
node2
dn
nomad
node3
nn
dn
nomad
node4
dn
nomad
node5

# Docker network

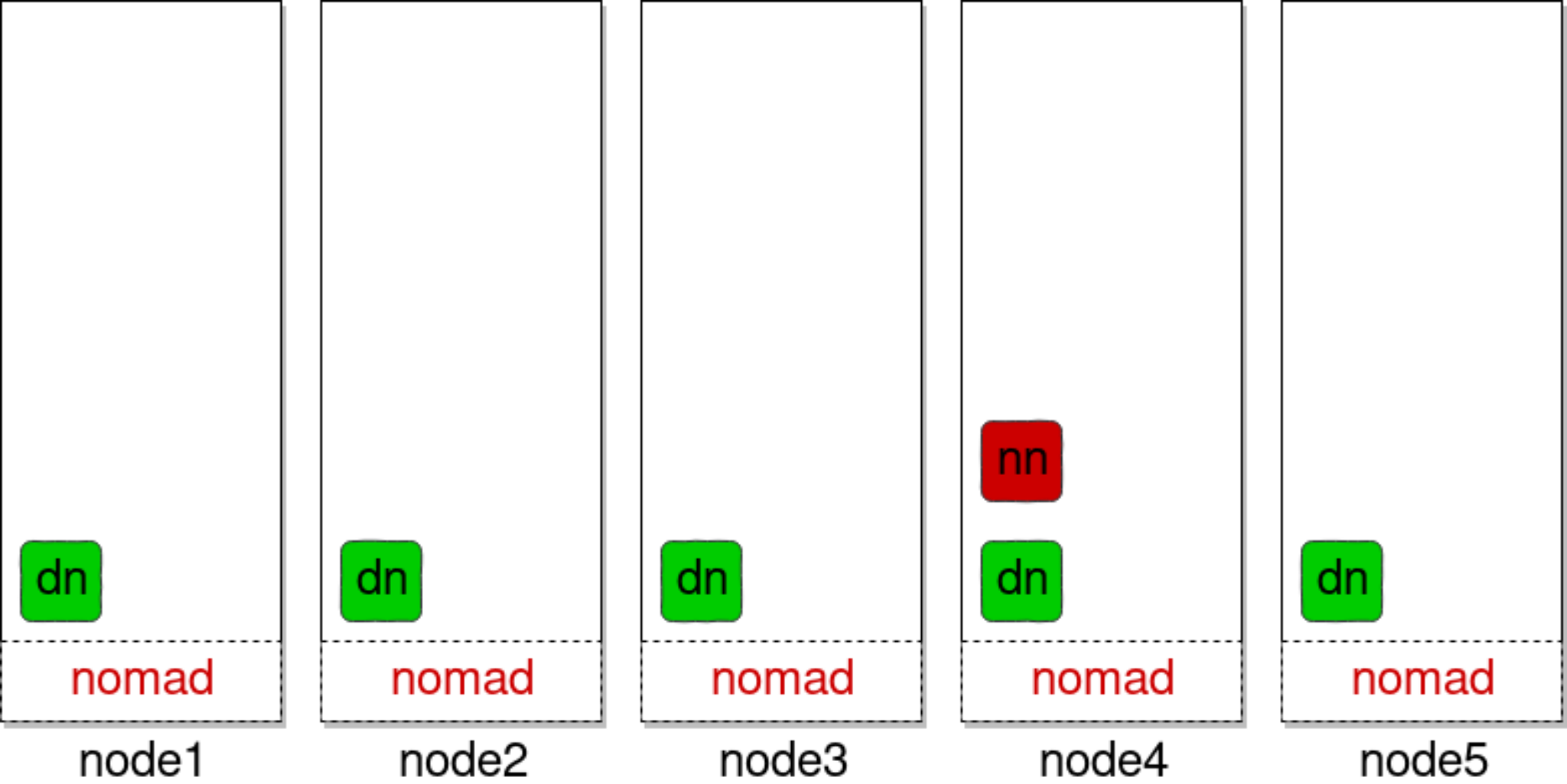


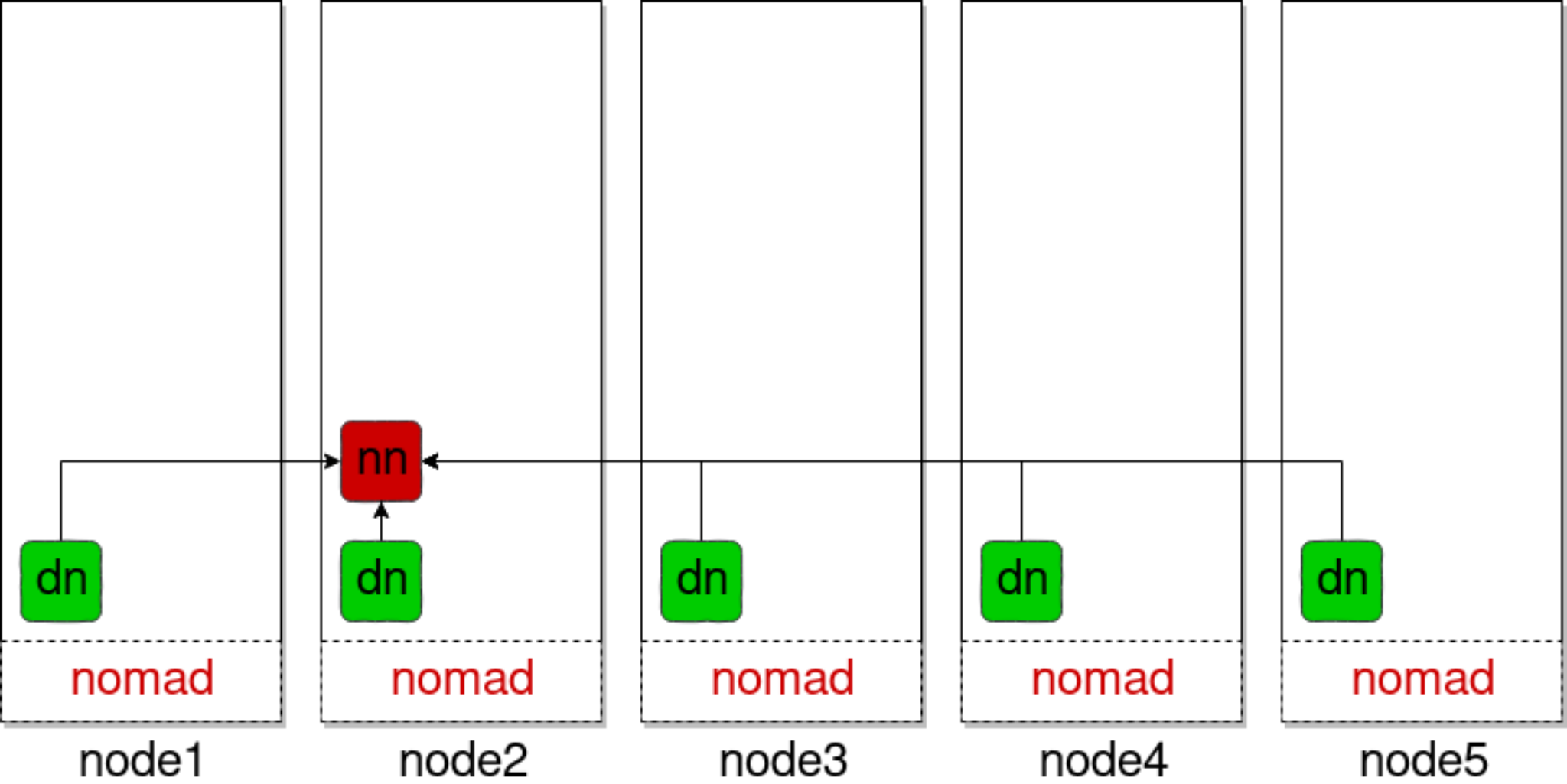
# Host network

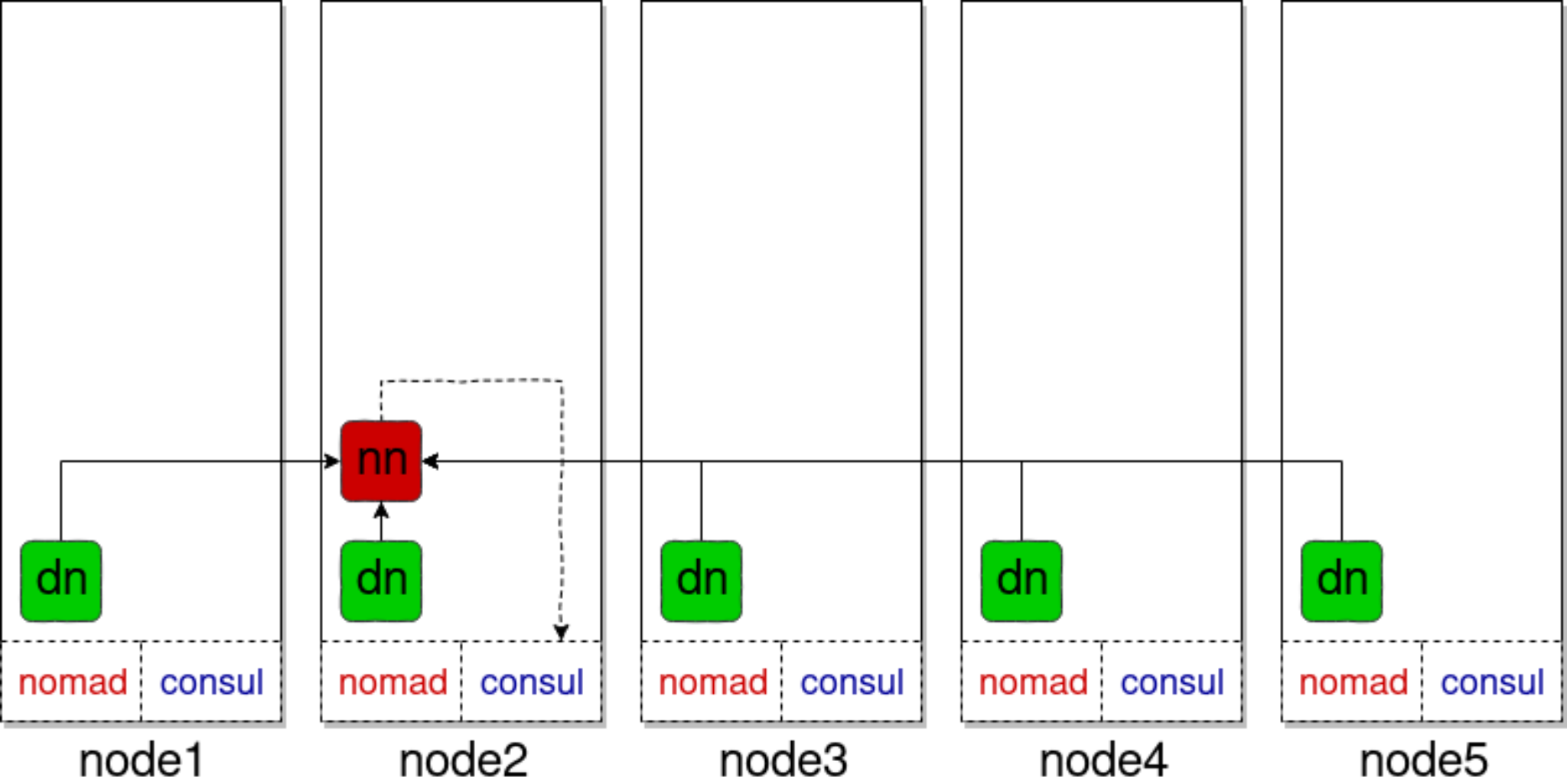


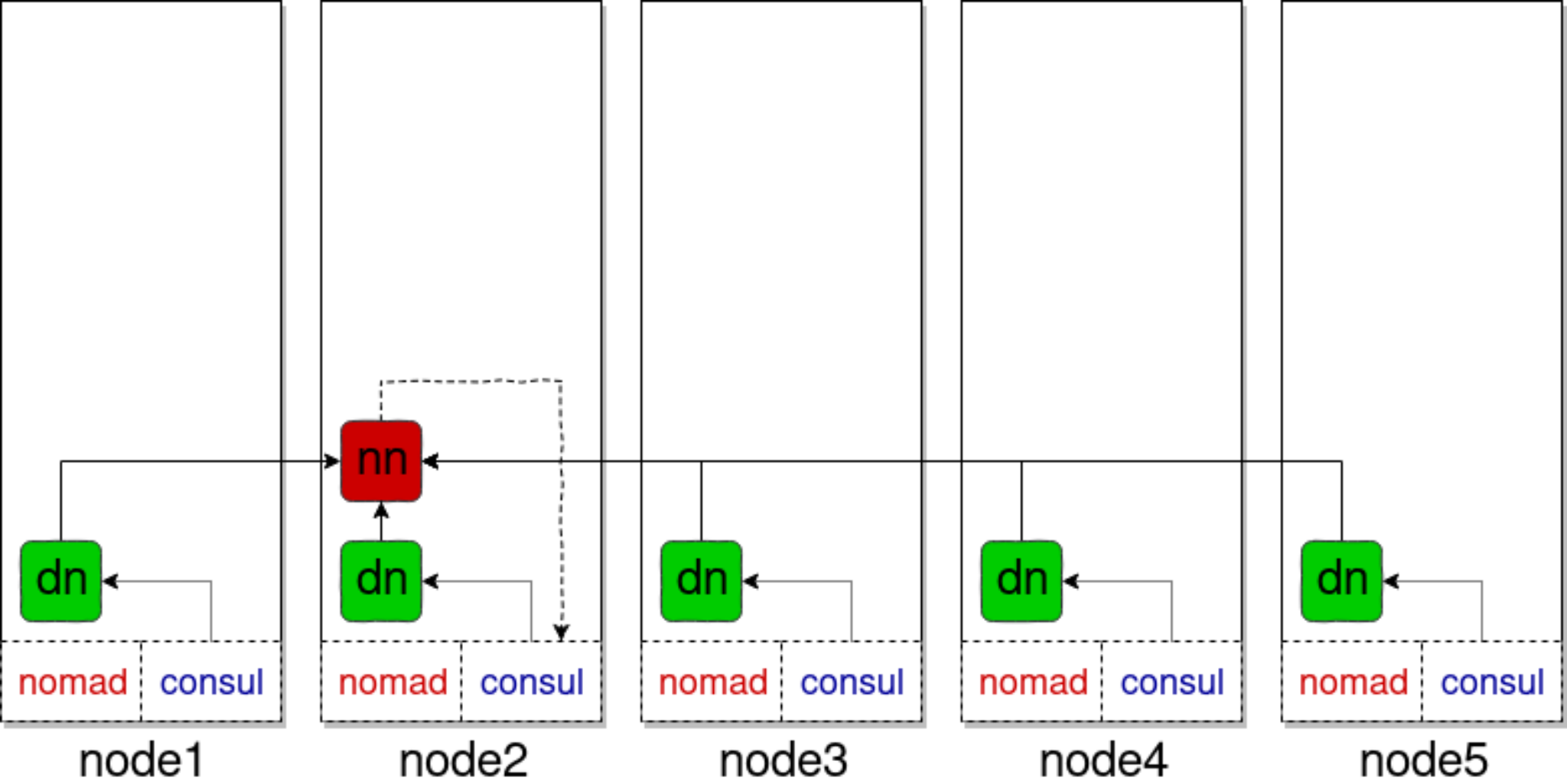
| | |
|---|---|
| **Configuration management** | |
| Source | |
| Preprocessing | |
| On change | |
| **Provisioning, Scheduling** | |
| Multihost support | |
| Scheduling | |
| Cluster definition | |
| Scaling | |
| Multi tenancy | |
| Failover | |
| **Network** | |
| Intraservice network | docker host |
| DNS | host dns |
| Service discovery | dns |
| Data locality | yes |
| Availability of the ports | host |

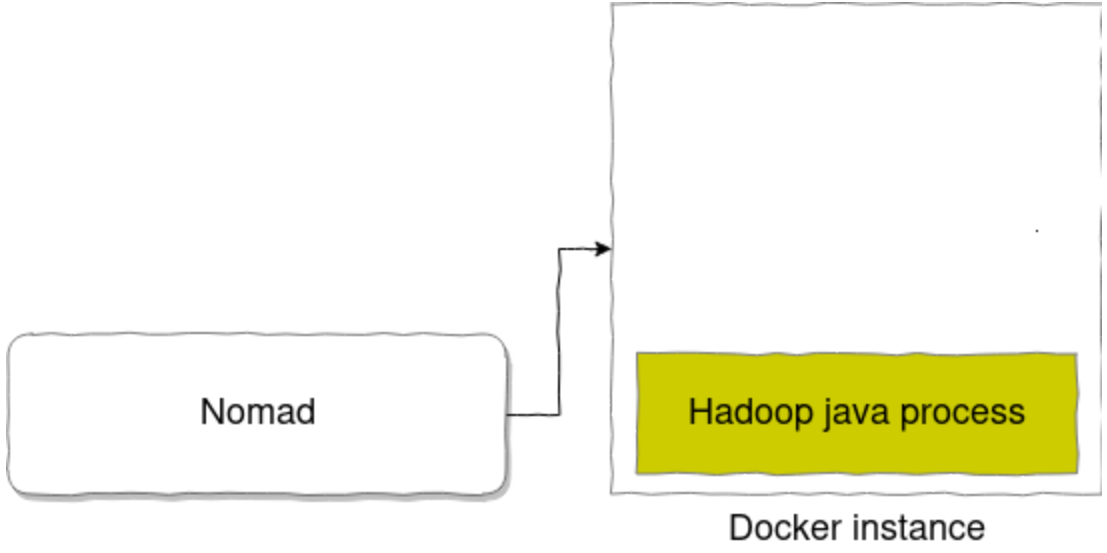
```
<configuration>
   <property>
      <name>dfs.namenode.rpc-address</name>
      <value>namenode:9000</value>
   </property>
   <property>
      <name>dfs.datanode.plugins</name>
      <value>org.apache.hadoop.ozone.HddsDatanodeService</value>
   </property>
   <property>
      <name>rpc.metrics.percentiles.intervals</name>
      <value>60,300</value>
   </property>
   <property>
      <name>dfs.namenode.name.dir</name>
      <value>/data/namenode</value>
   </property>
   <property>
      <name>rpc.metrics.quantile.enable</name>
      <value>true</value>
   </property>
</configuration>
```

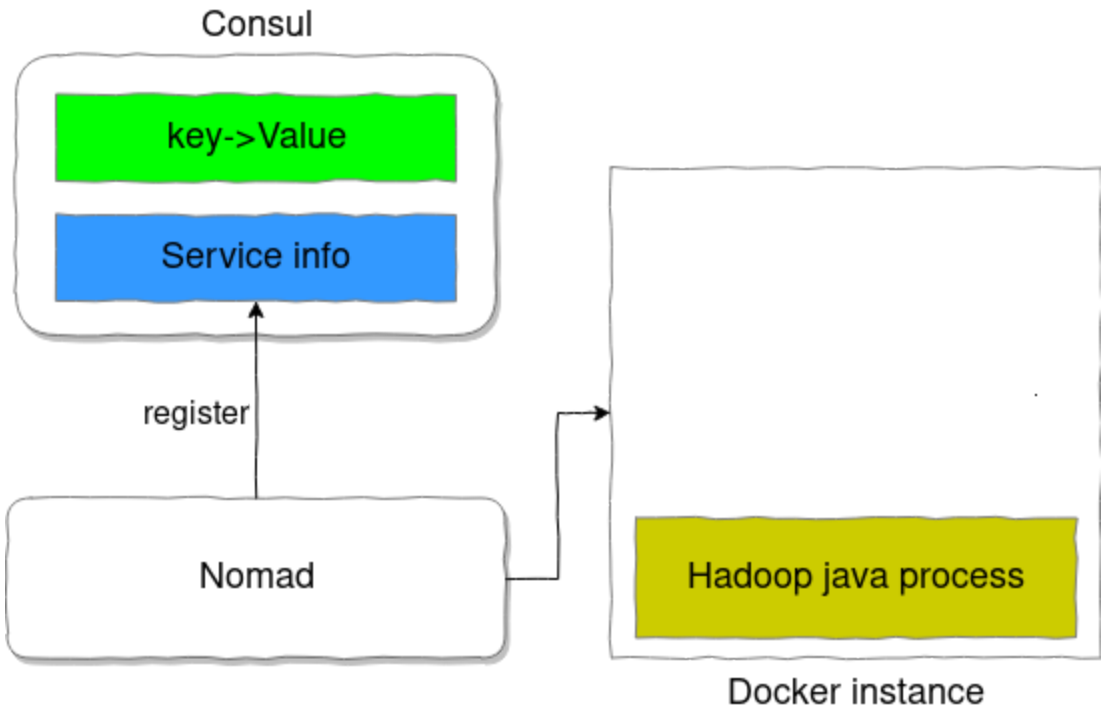

dn
nomad
node1
dn
nomad
node2
dn
nomad
node3
nn
dn
nomad
node4
dn
nomad
node5

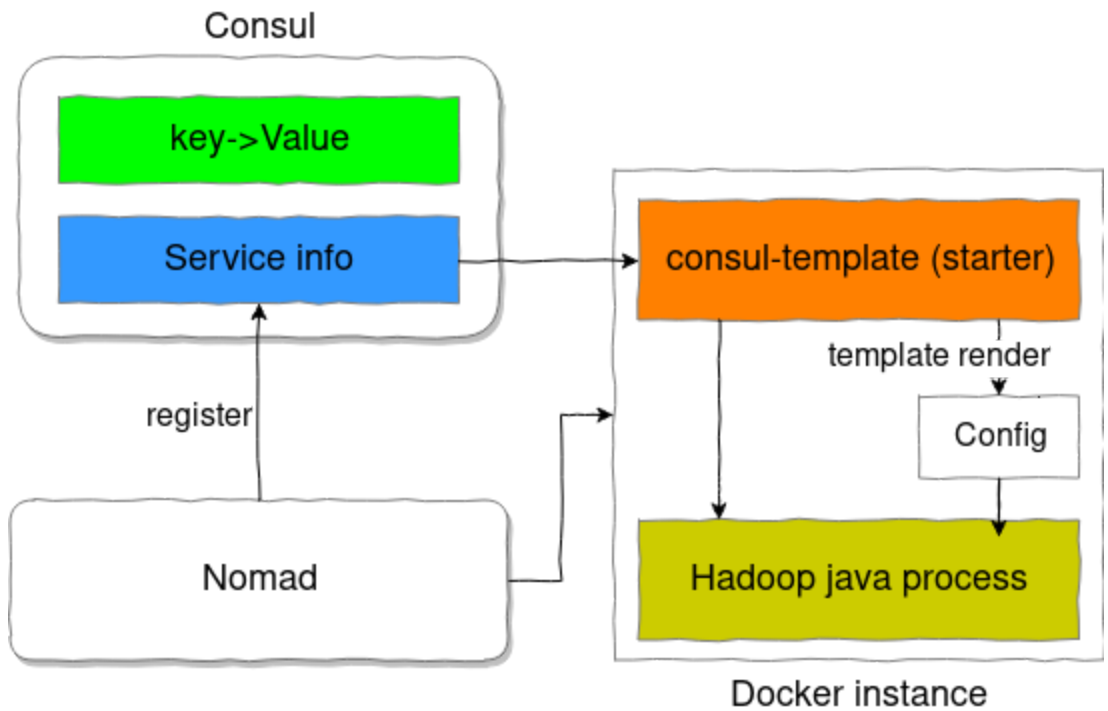

nn
dn
dn
dn
dn
dn
nomad
nomad
nomad
nomad
nomad
node1
node2
node3
node4
node5

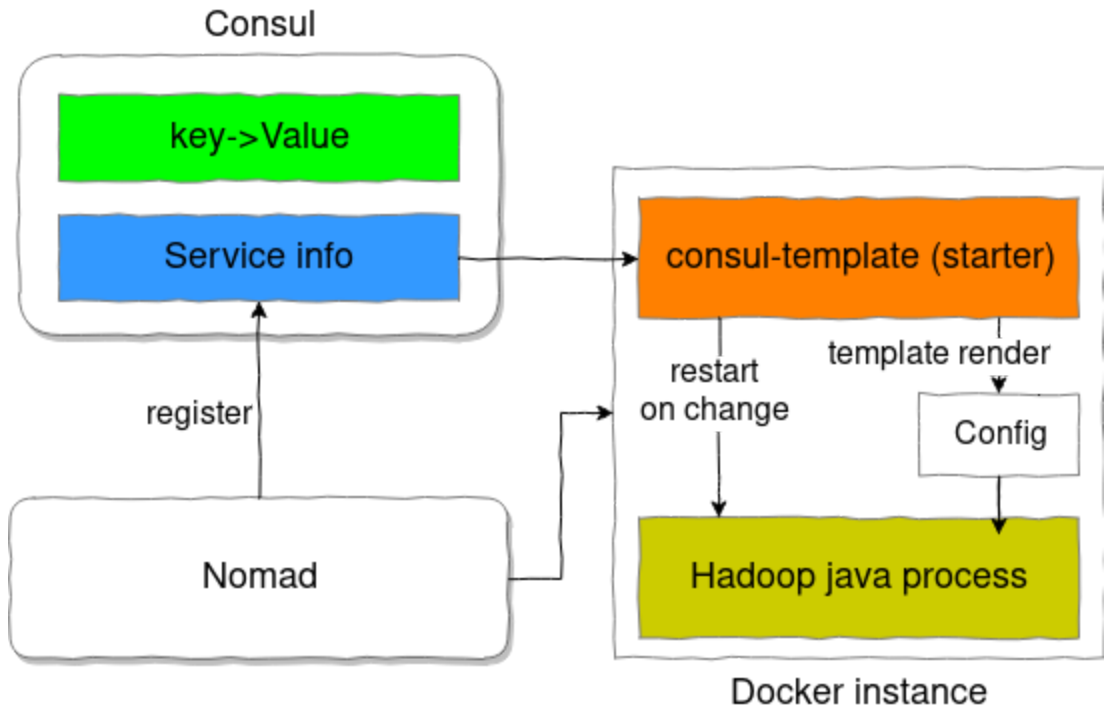

nn
dn
dn
dn
dn
dn
nomad
consul
nomad
consul
nomad
consul
nomad
consul
nomad
consul
node1
node2
node3
node4
node5

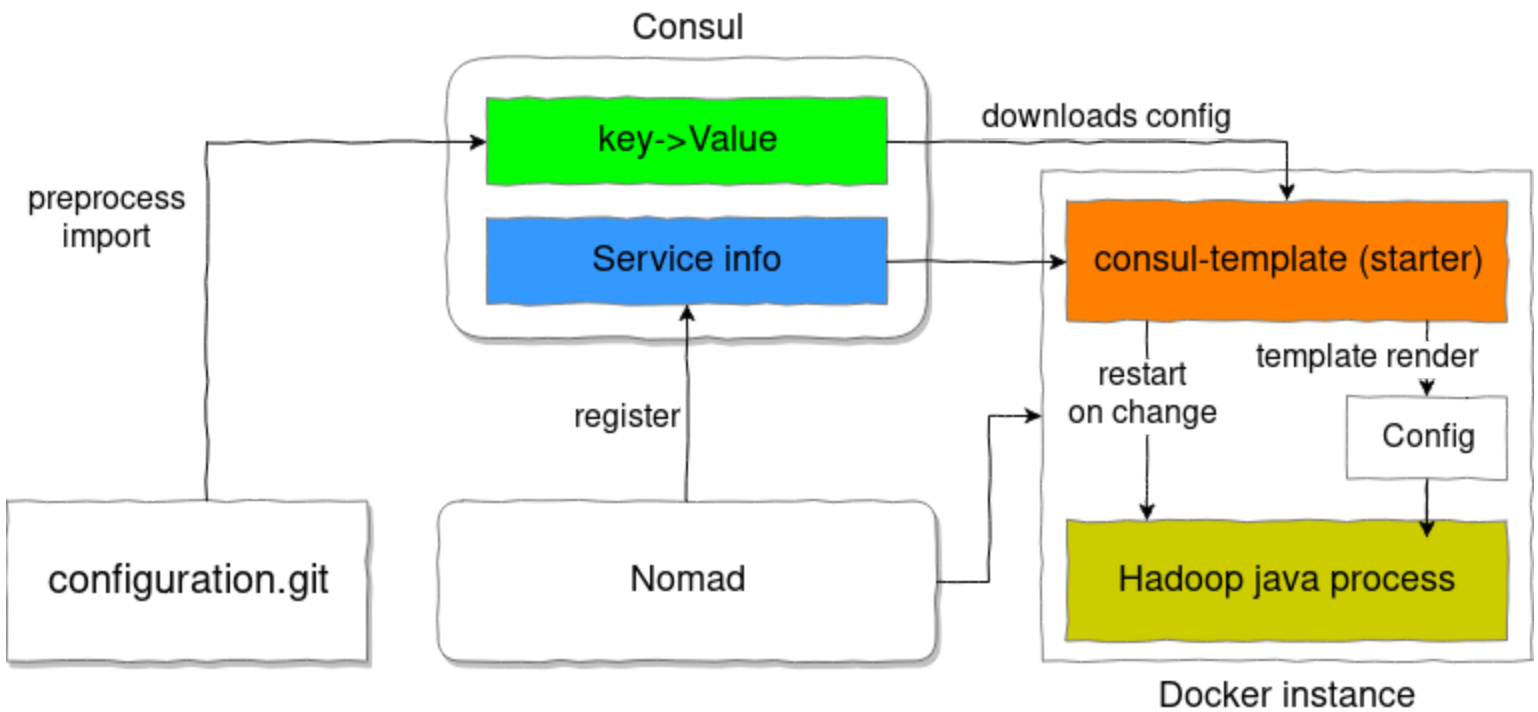

nn
dn
dn
dn
dn
dn
nomad
consul
nomad
consul
nomad
consul
nomad
consul
nomad
consul
node1
node2
node3
node4
node5


Nomad
Hadoop java process
Docker instance

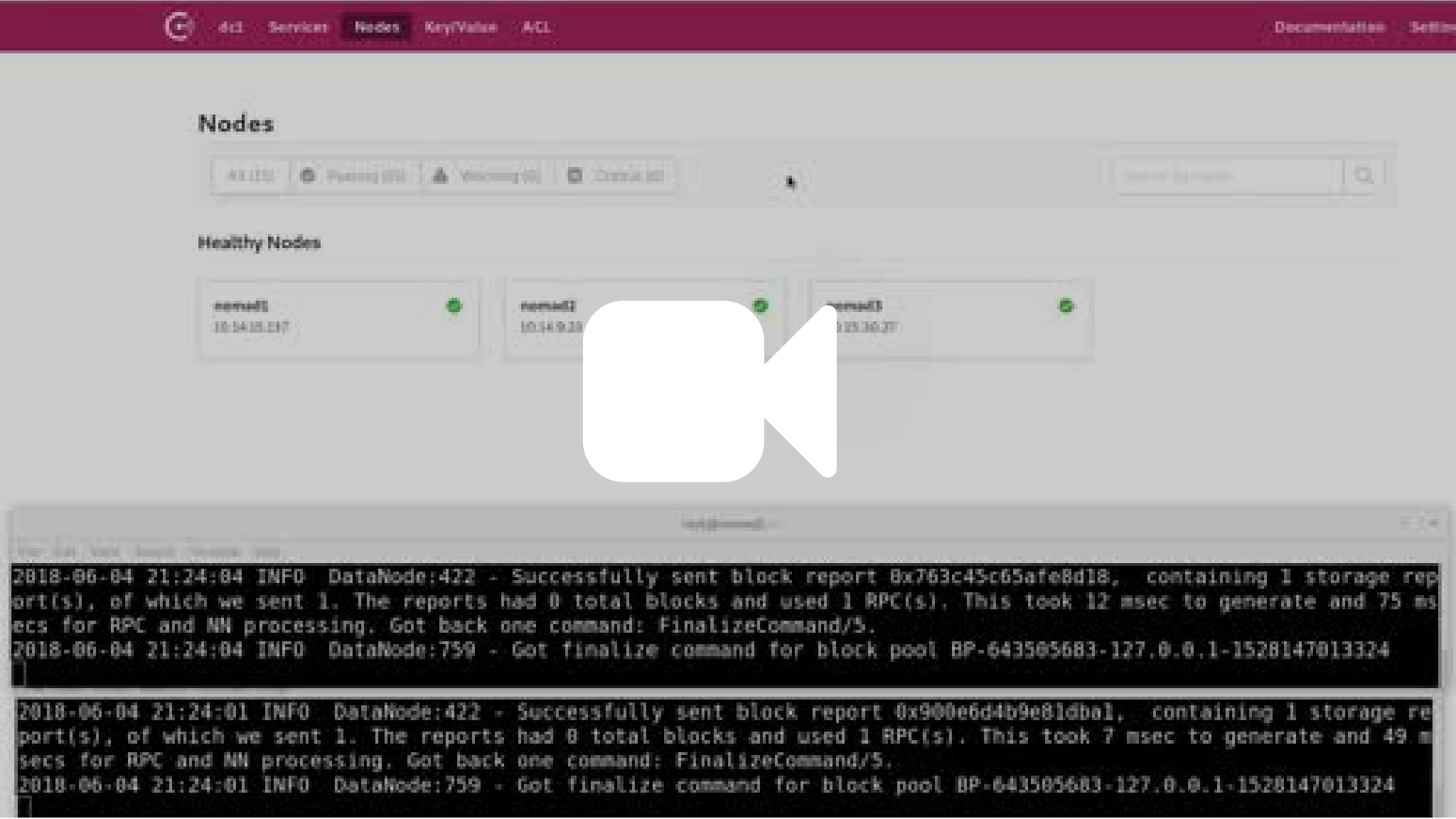

Consul
key->Value
Service info
register
Nomad
Hadoop java process
Docker instance


Consul
key->Value
Service info
consul-template (starter)
template render
Config
register
Nomad
Hadoop java process
Docker instance

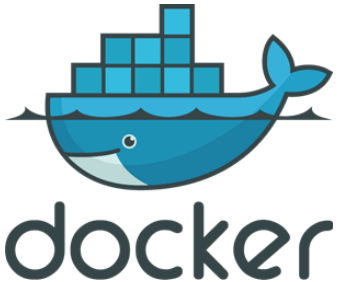

Consul
key->Value
Service info
register
Nomad
consul-template (starter)
restart
on change
template render
Config
Hadoop java process
Docker instance


Consul
key->Value
Service info
downloads config
preprocess
import
consul-template (starter)
restart
on change
template render
Config
register
configuration.git
Nomad
Hadoop java process
Docker instance


# DIY

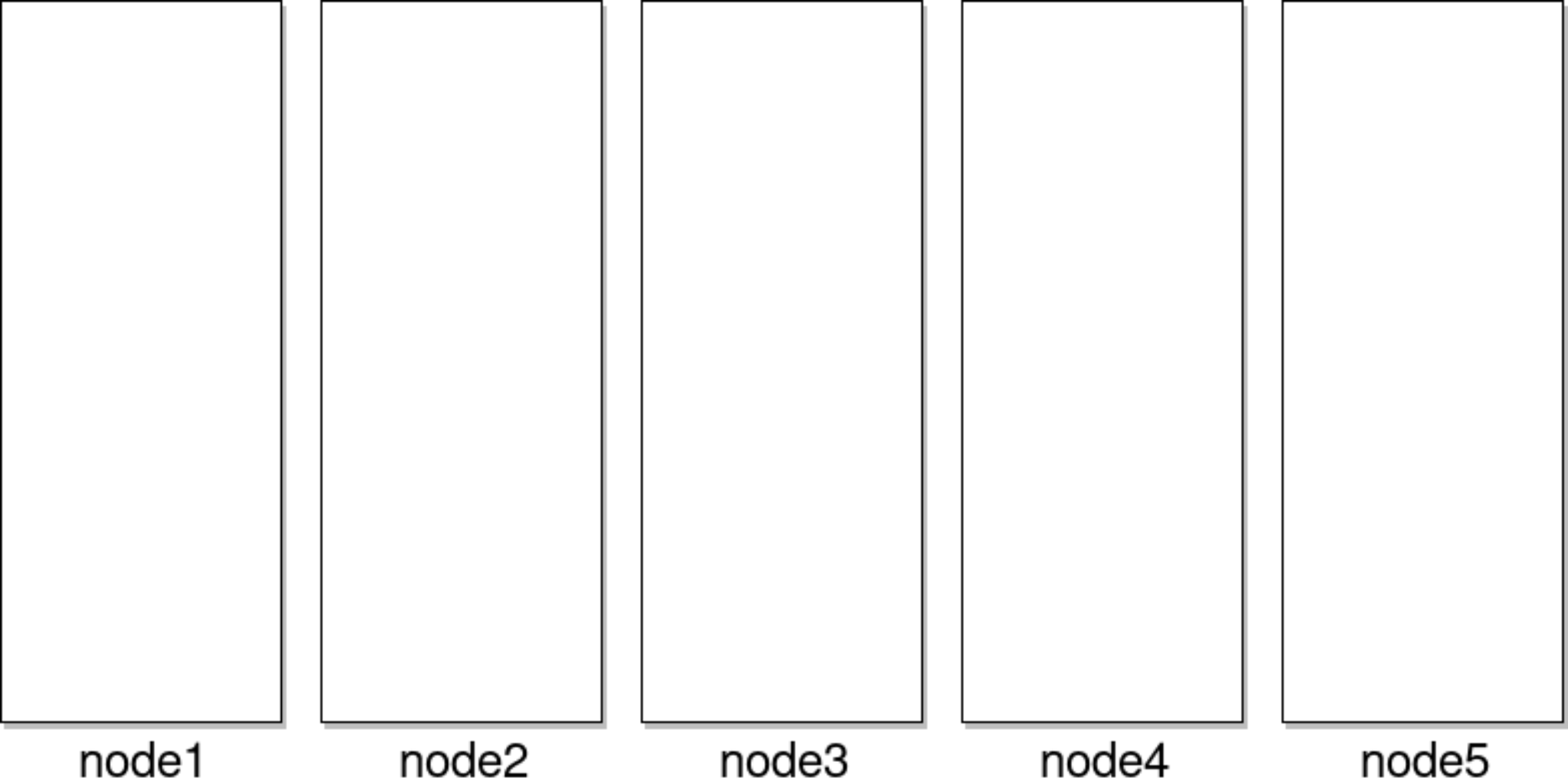

dc1
Services
Nodes
Key/Value
ACL
Documentation
Nodes
Healthy Nodes
nomad1
nomad2
nomad3
2018-06-04 21:24:04 INFO DataNode:422 - Successfully sent block report 0x763c45c65afe8d18, containing 1 storage report(s), of which we sent 1. The reports had 0 total blocks and used 1 RPC(s). This took 12 msec to generate and 75 msecs for RPC and NN processing. Got back one command: FinalizeCommand/5.
2018-06-04 21:24:04 INFO DataNode:759 - Got finalize command for block pool BP-643505683-127.0.0.1-1528147013324
2018-06-04 21:24:01 INFO DataNode:422 - Successfully sent block report 0x900e6d4b9e81dba1, containing 1 storage report(s), of which we sent 1. The reports had 0 total blocks and used 1 RPC(s). This took 7 msec to generate and 49 msecs for RPC and NN processing. Got back one command: FinalizeCommand/5.
2018-06-04 21:24:01 INFO DataNode:759 - Got finalize command for block pool BP-643505683-127.0.0.1-1528147013324

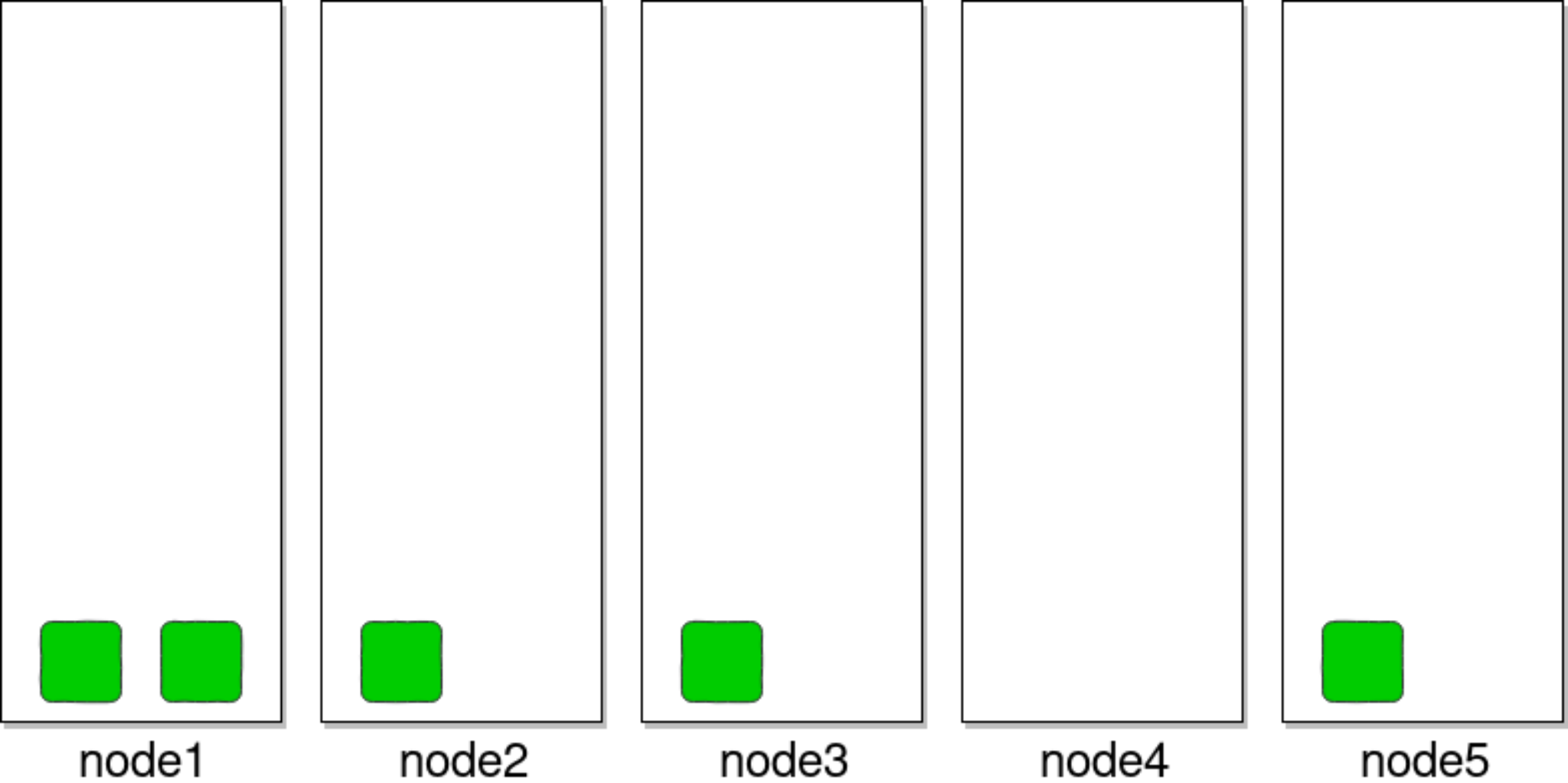

| | |
|---|---|
| **Configuration management** | |
| Source | Consul |
| Preprocessing | Yes (script) |
| On change | Restart |
| **Provisioning, Scheduling** | |
| Multihost support | host netw |
| Scheduling | Nomad |
| Cluster definition | .nomad |
| Scaling | redeploy |
| Multi tenancy | no |
| Failover | yes |
| **Network** | |
| Intraservice network | host netw |
| DNS | yes |
| Service discovery | consul |
| Data locality | yes |
| Availability of the ports | host |

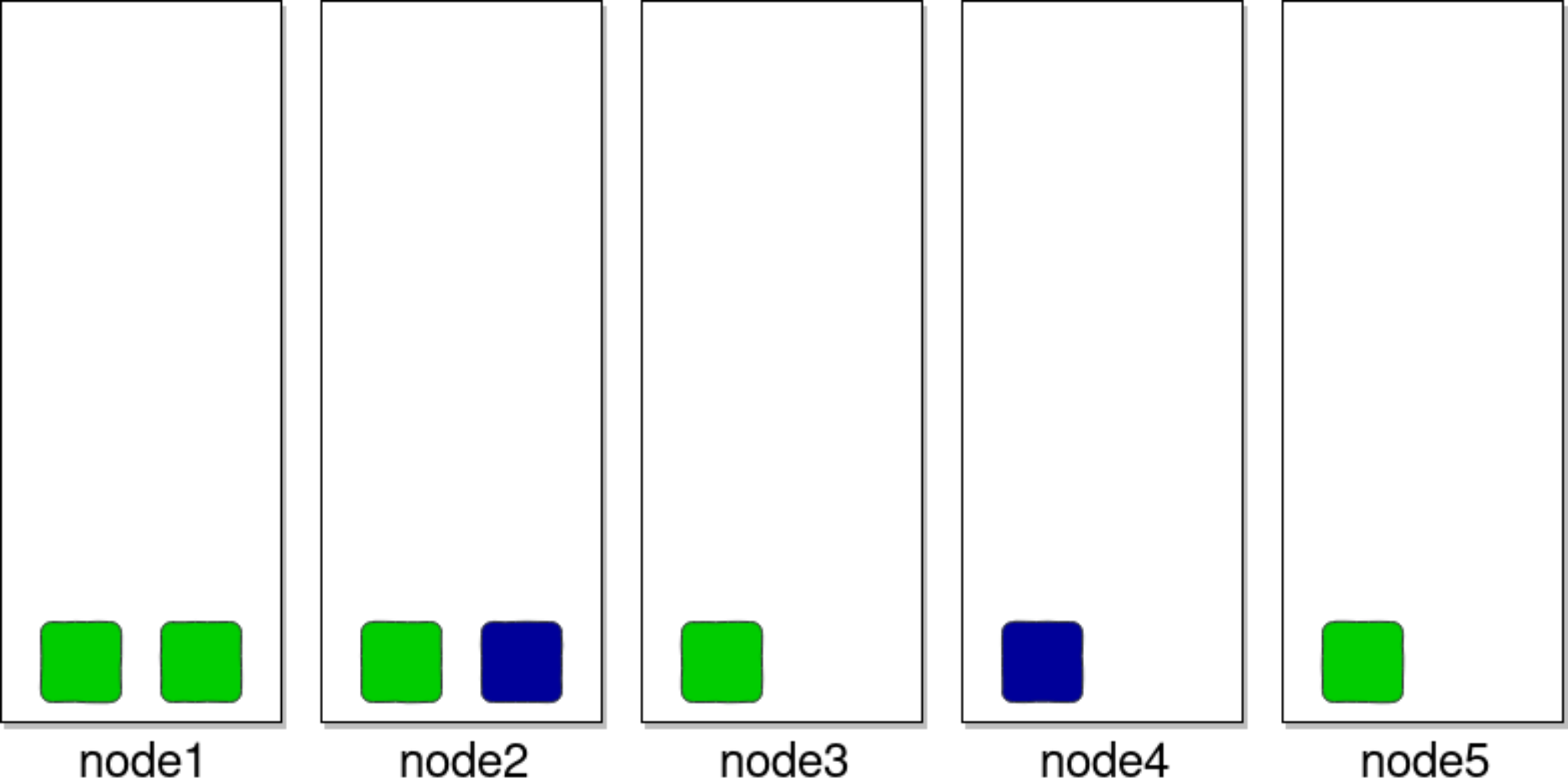

docker

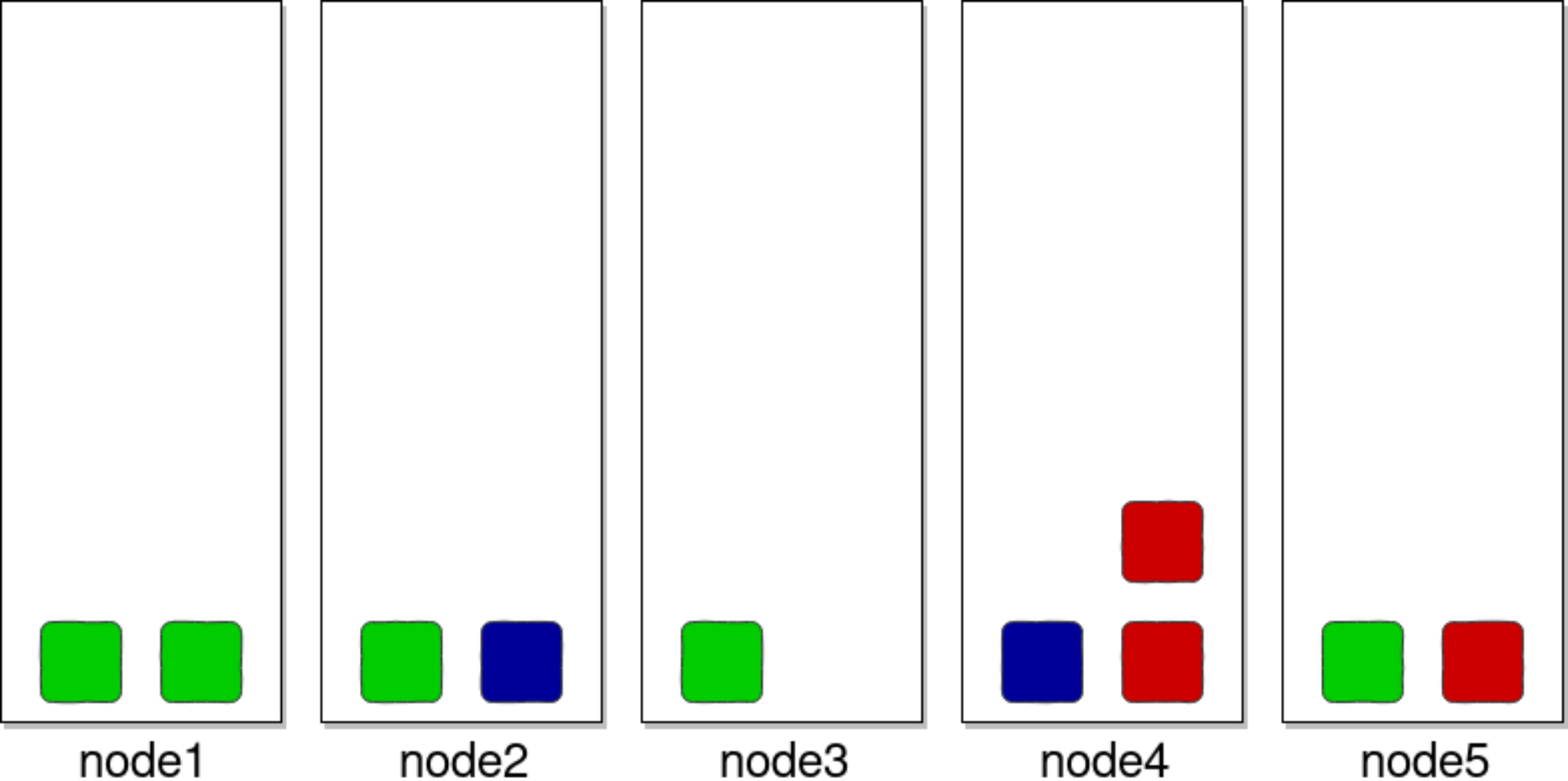
# Kubernetes

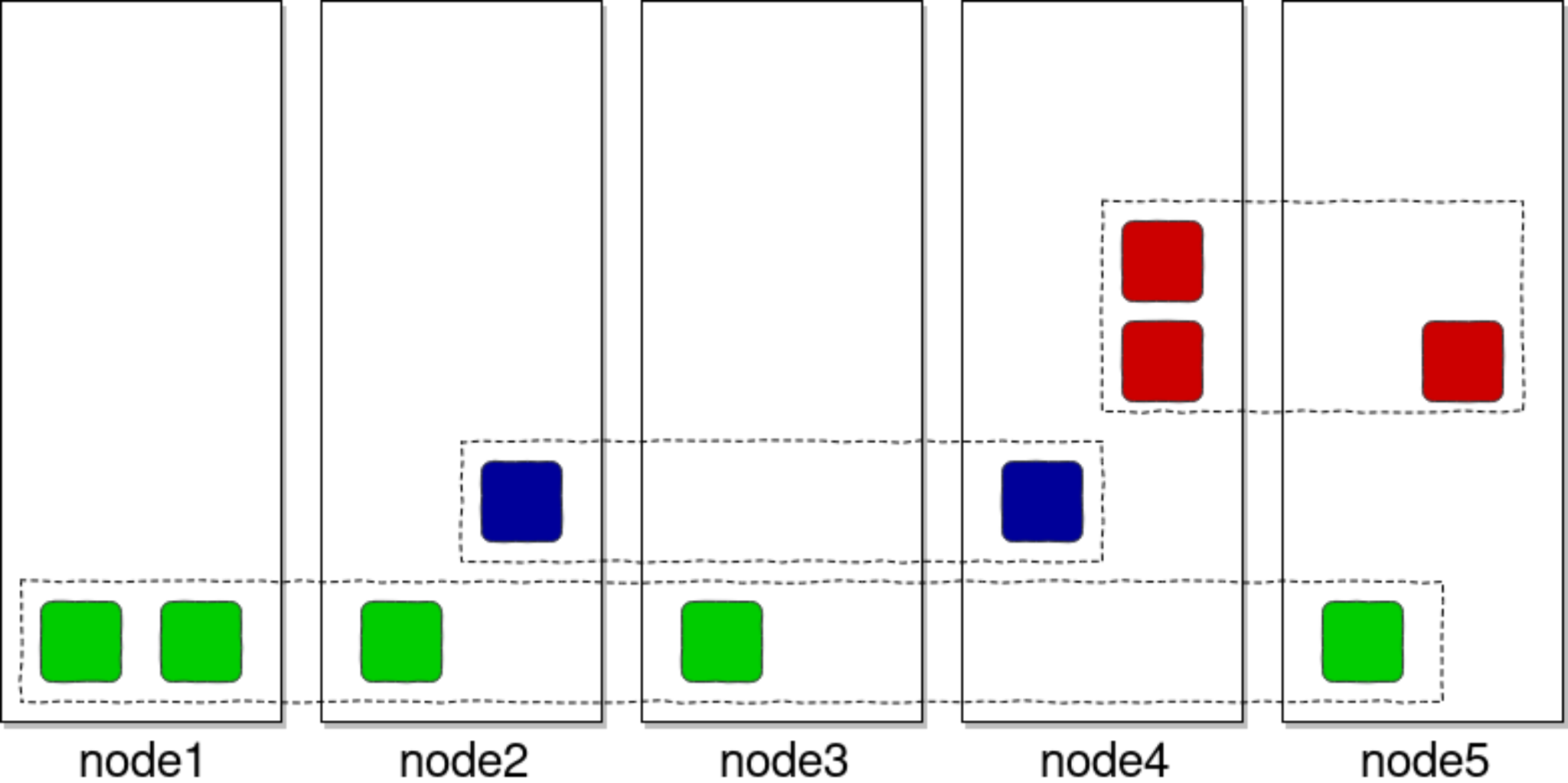
"out of the box"


node1
node2
node3
node4
node5


node1
node2
node3
node4
node5

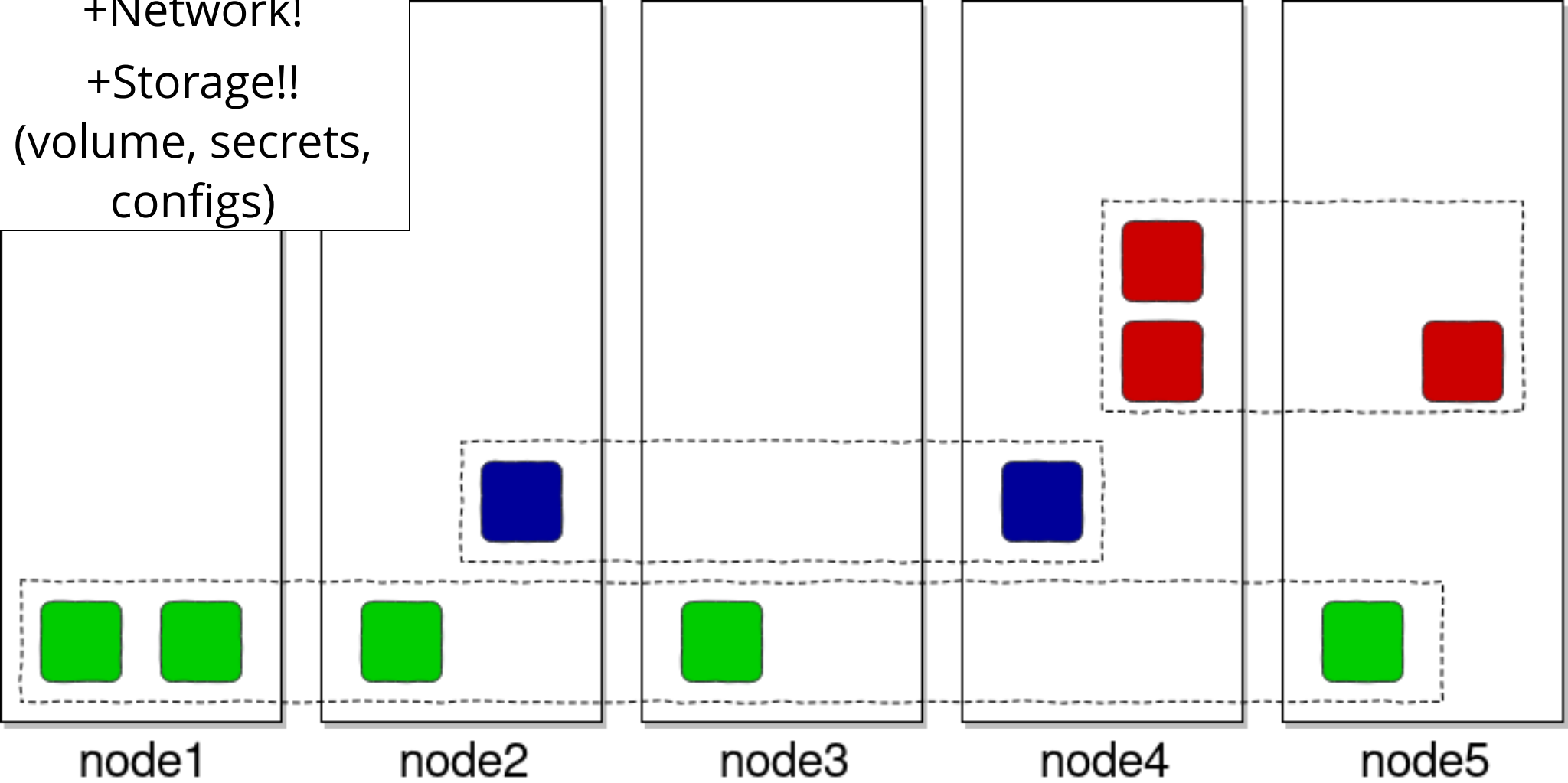

node1
node2
node3
node4
node5

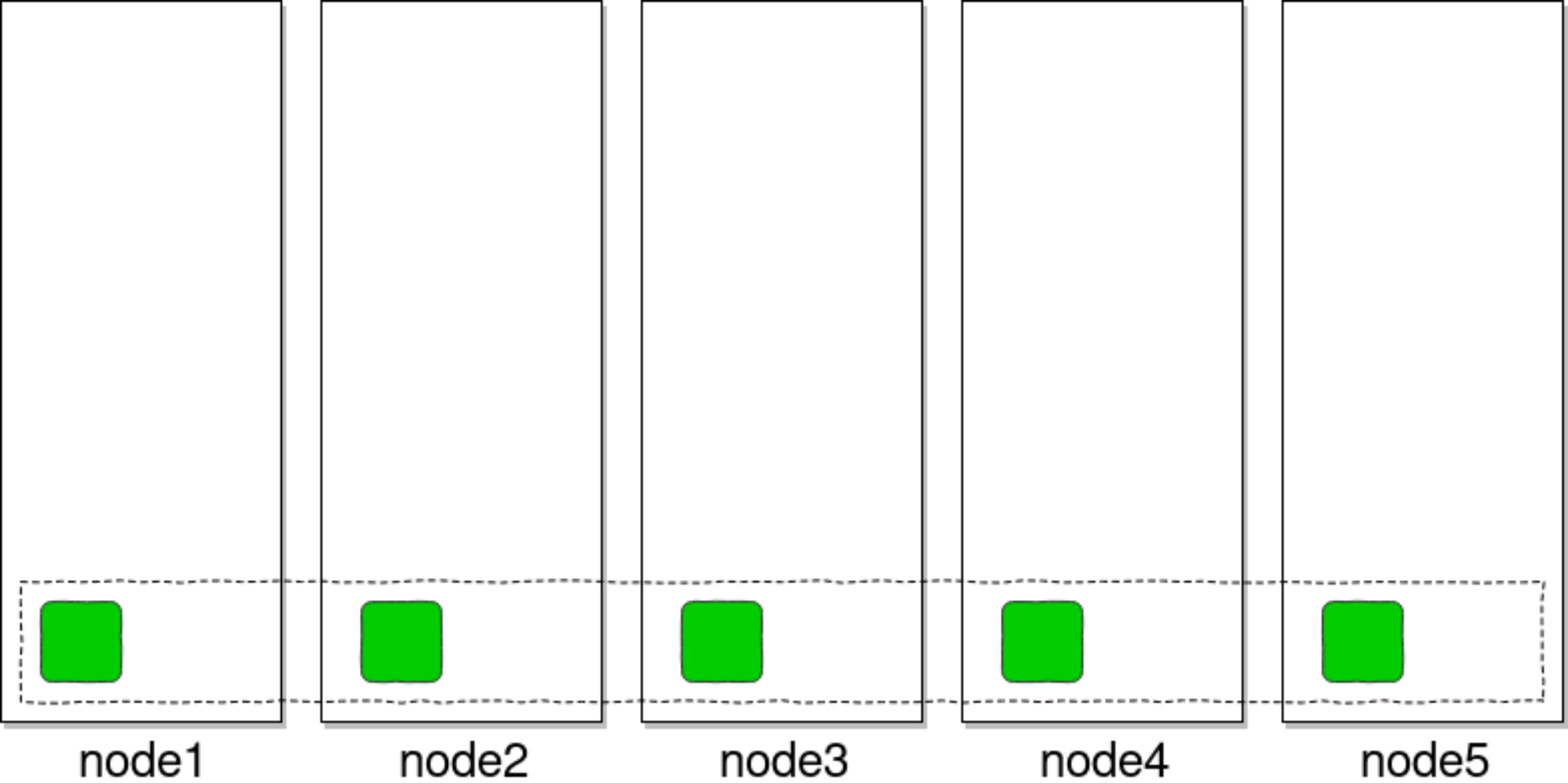

node1
node2
node3
node4
node5


node1
node2
node3
node4
node5

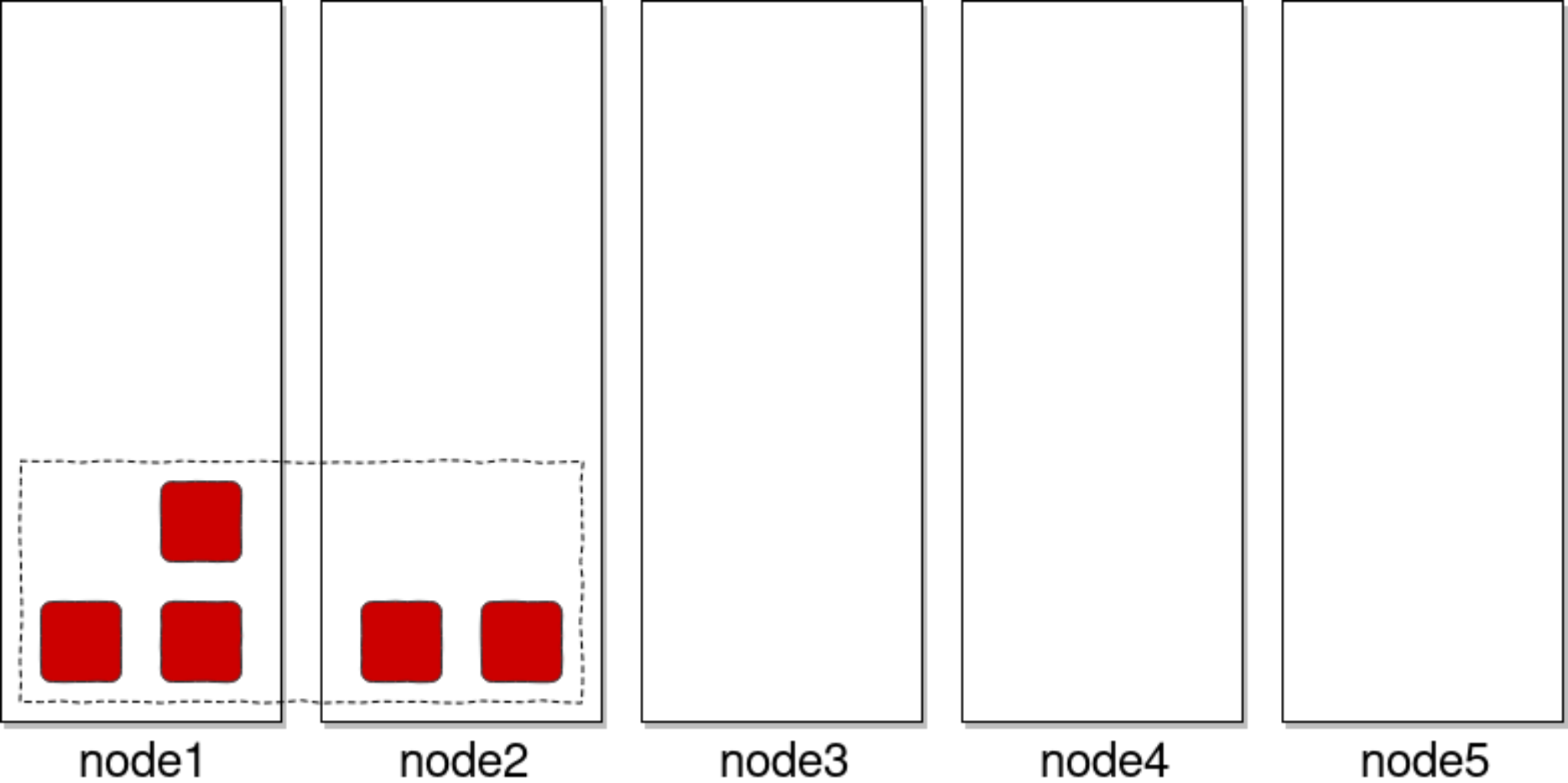
+Network!

+Storage!!
(volume, secrets, configs)






DaemonSet

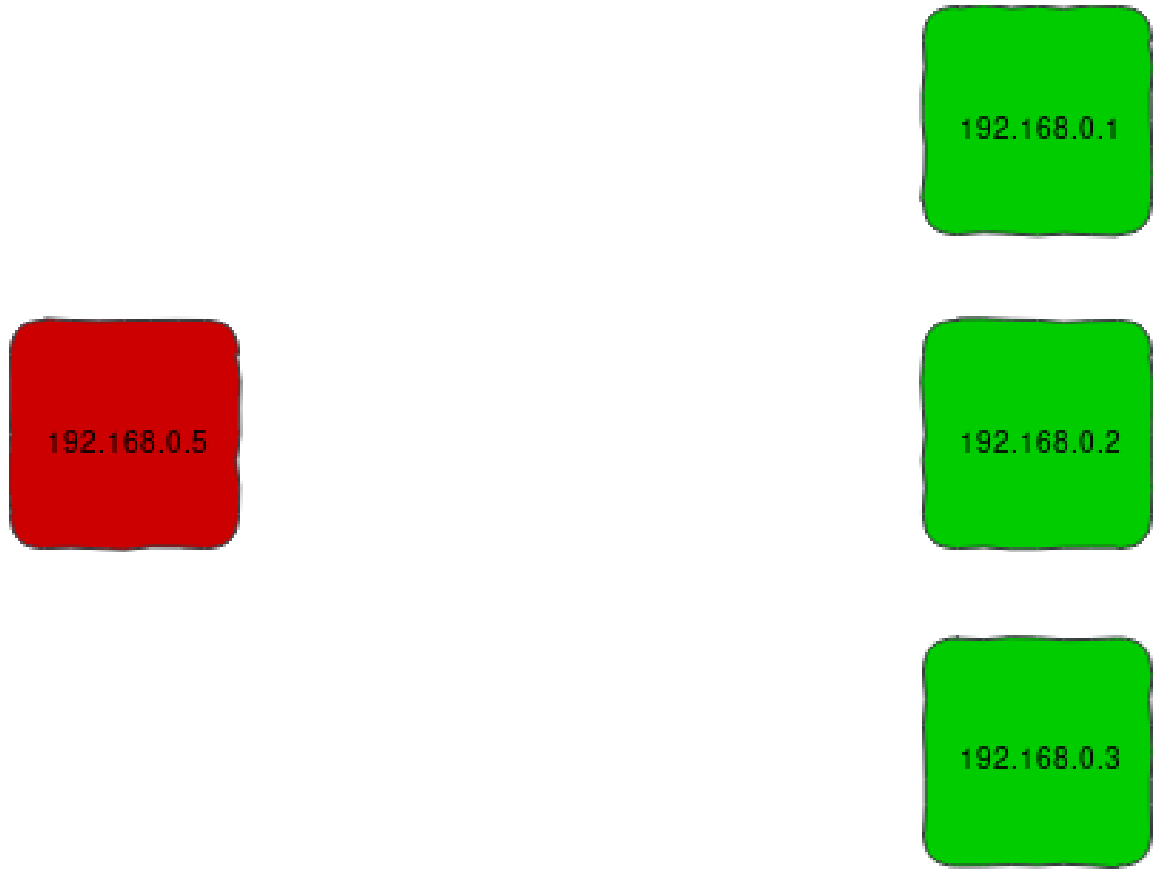



ReplicaSet

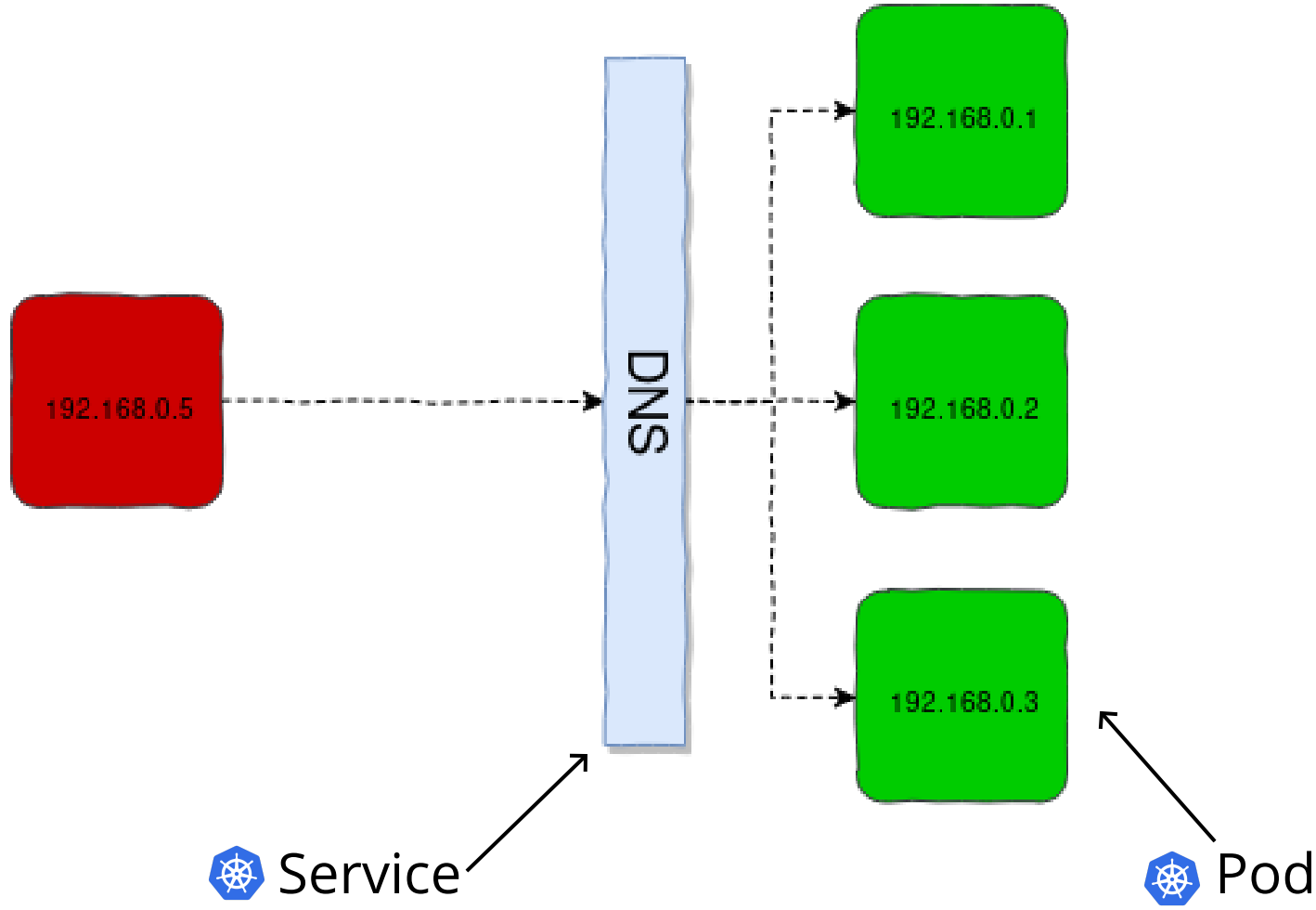
192.168.0.1

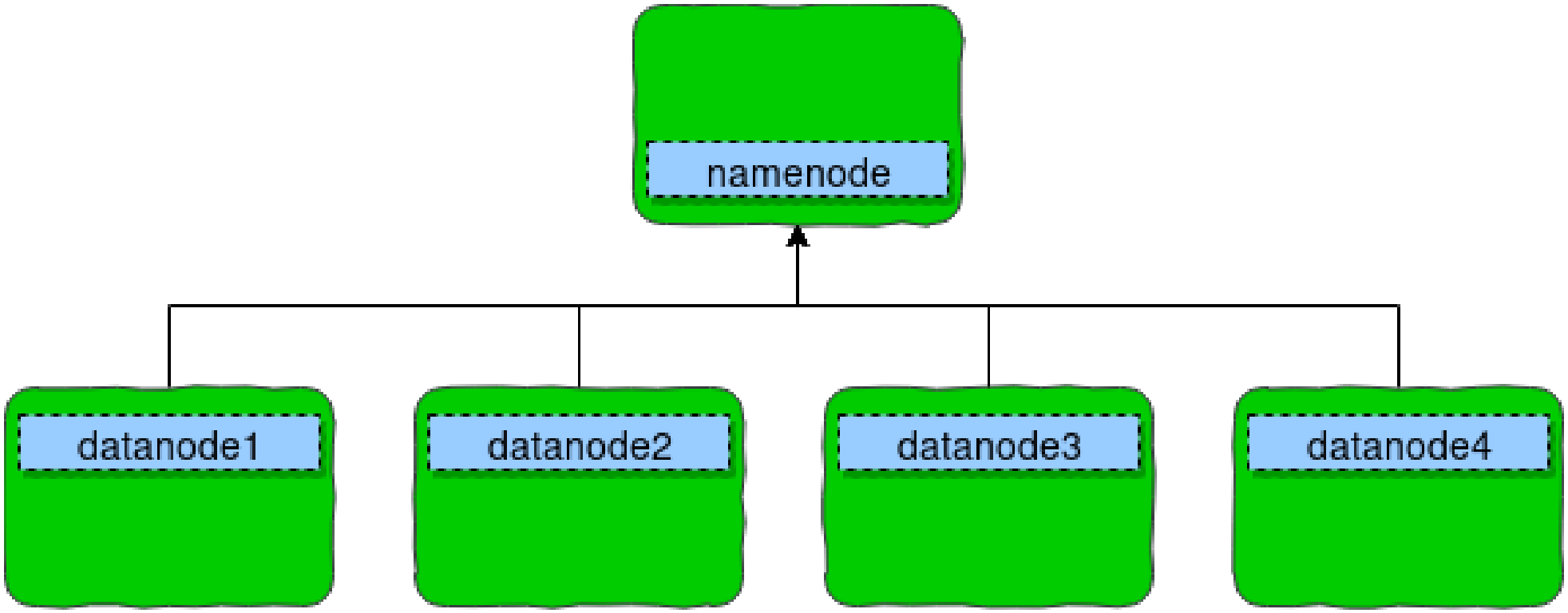
192.168.0.2

192.168.0.3

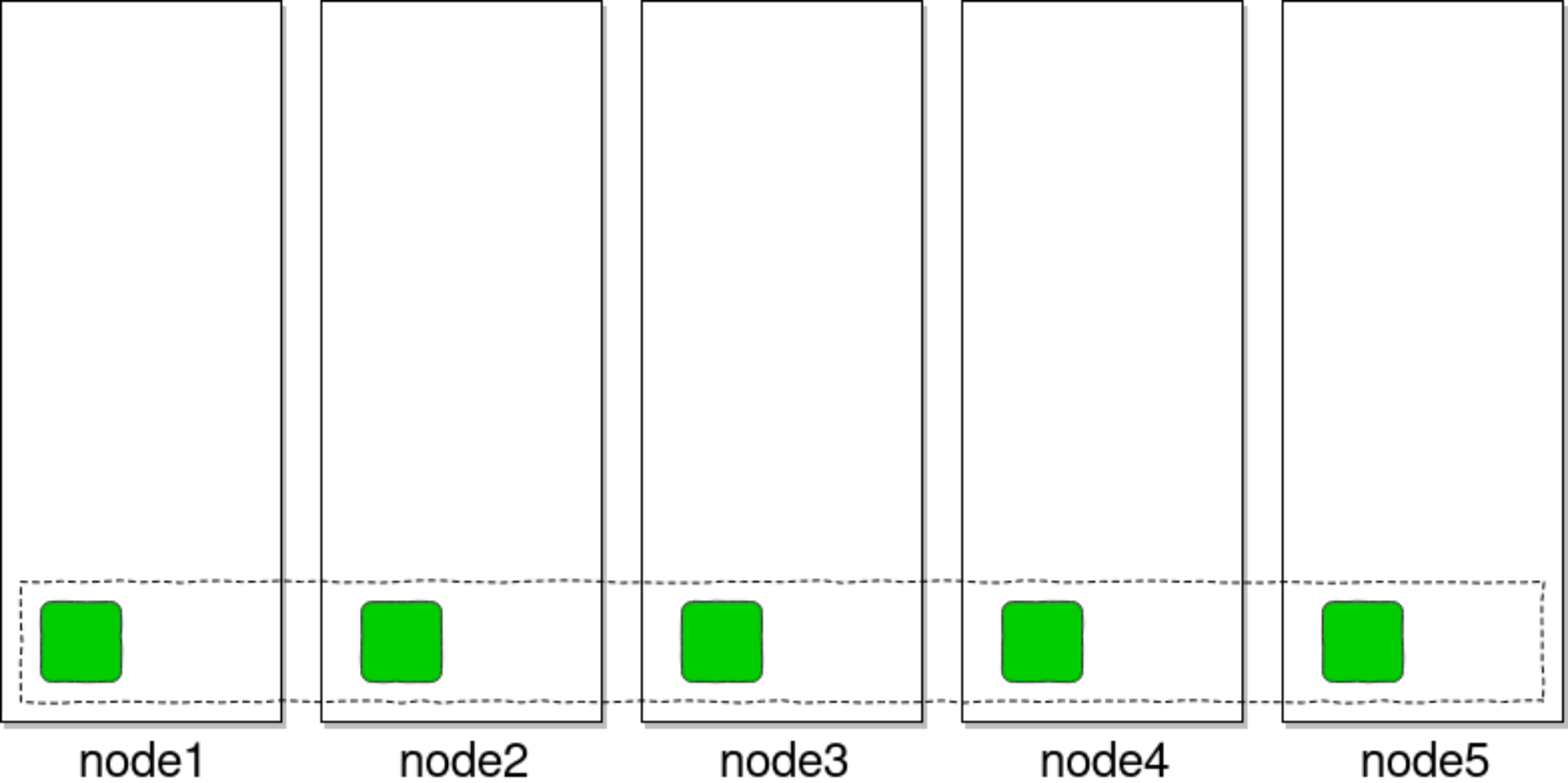

192.168.0.1
192.168.0.5
192.168.0.2
192.168.0.3

192.168.0.5
DNS
192.168.0.1
192.168.0.2
192.168.0.3

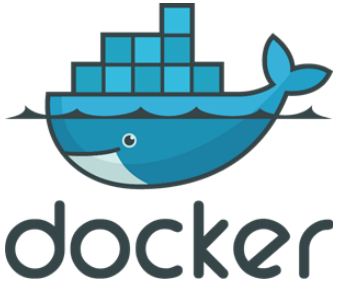

192.168.0.5
DNS
192.168.0.1
192.168.0.2
192.168.0.3
Service
Pod



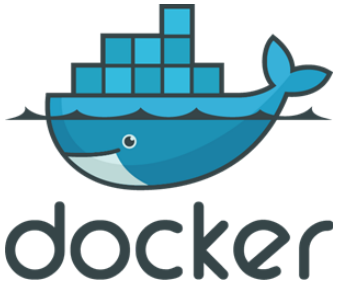

StatefulSet




DaemonSet


docker

192.168.0.5
DNS
192.168.0.1
192.168.0.2
192.168.0.3


docker

# Benefits of Hadoop + k8s?

# Benefits of Hadoop + k8s?

Ecosystem

Flexibility

# Example:

## Monitor Hadoop with Prometheus

Prometheus Alerts Graph Status Help

Enable query history

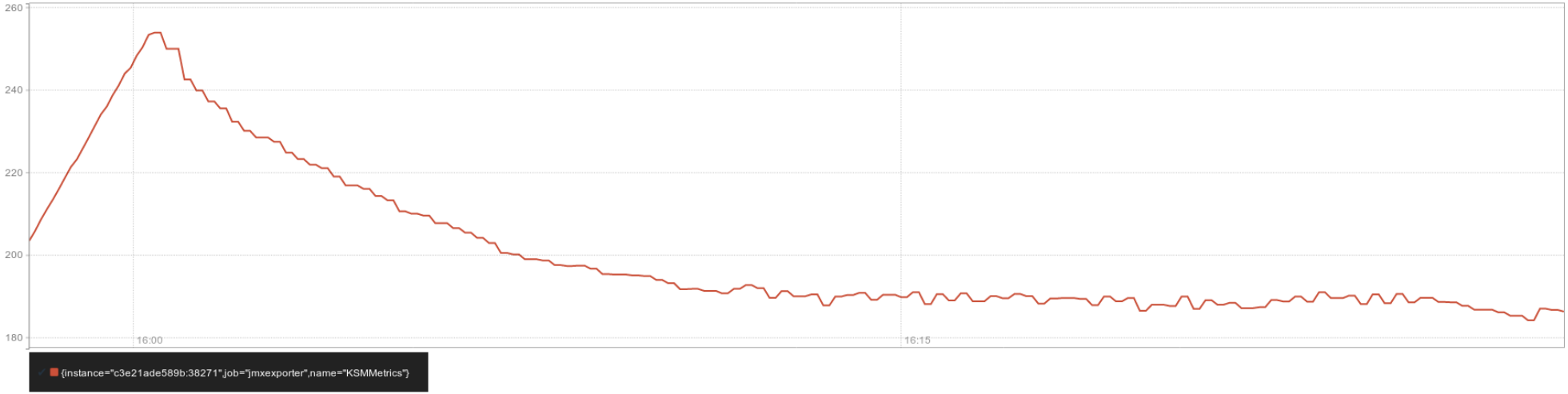
rate(Hadoop_KeySpaceManager_NumKeyCommits[10m])

Load time: 187ms
Resolution: 7s
Total time series:

Execute java_lang_OperatingSyste

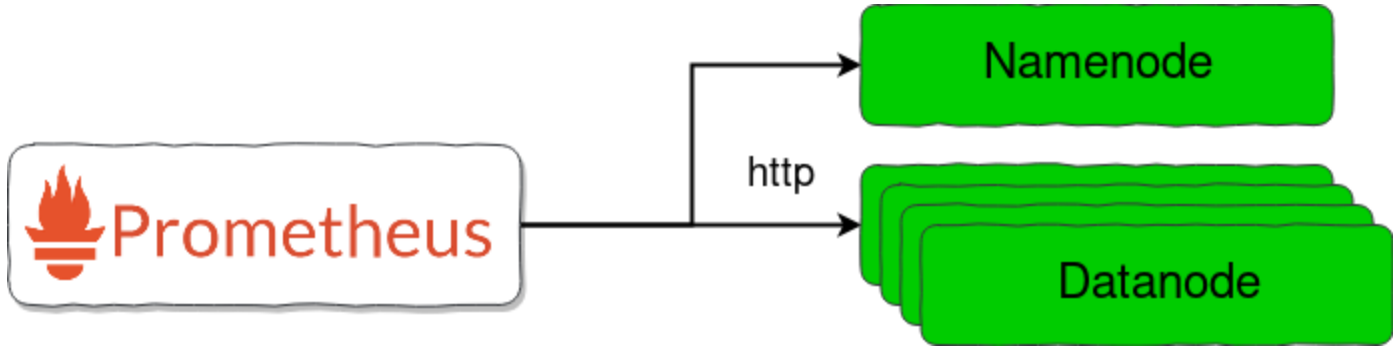
Graph Console

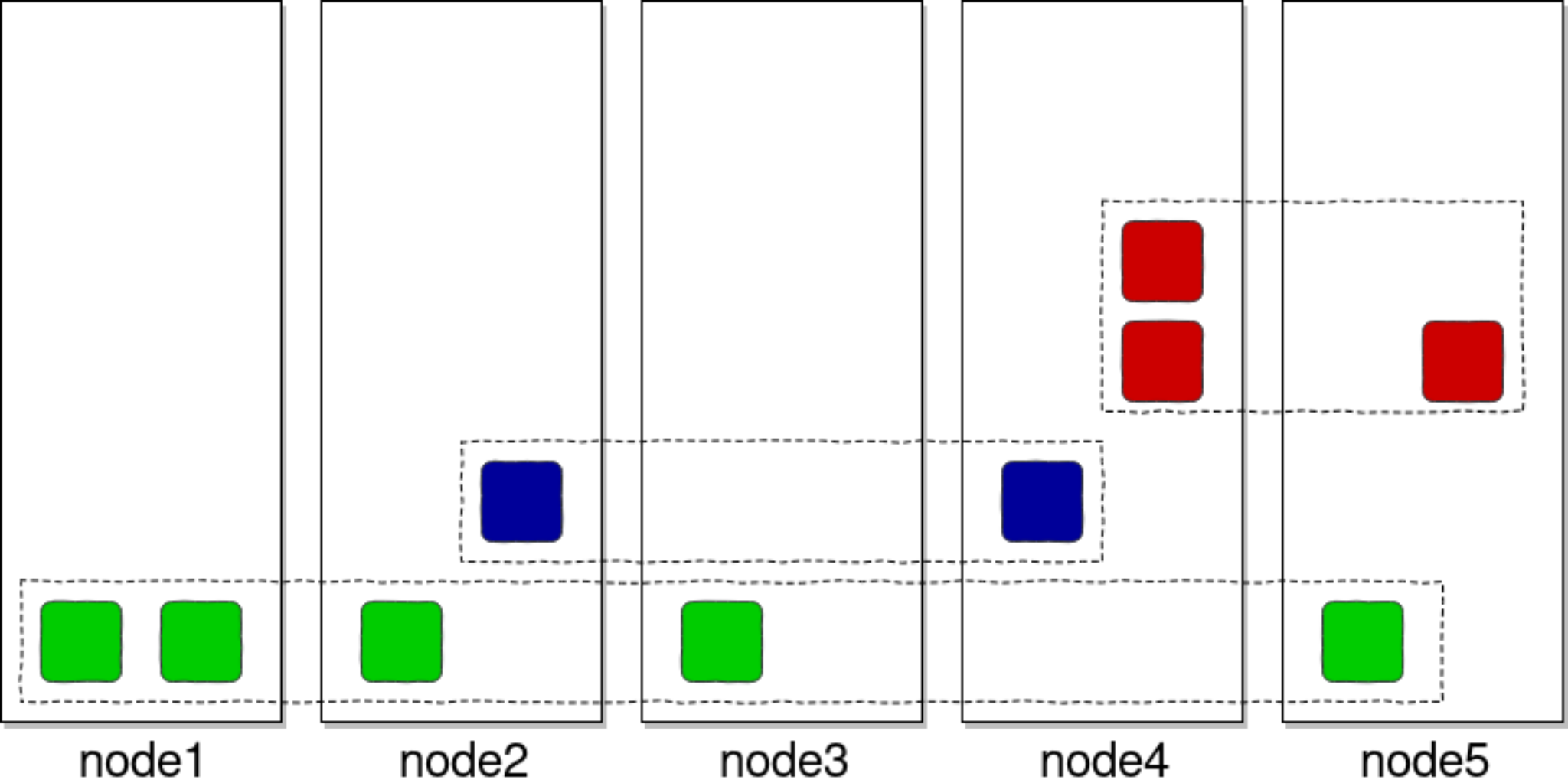
30m Until Res. (s) stacked

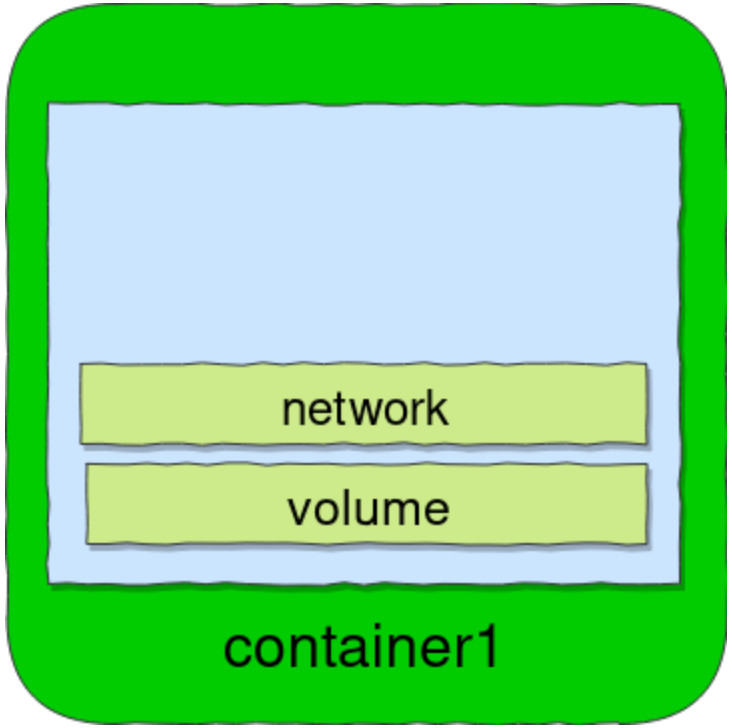



Remove Grap

Add Graph

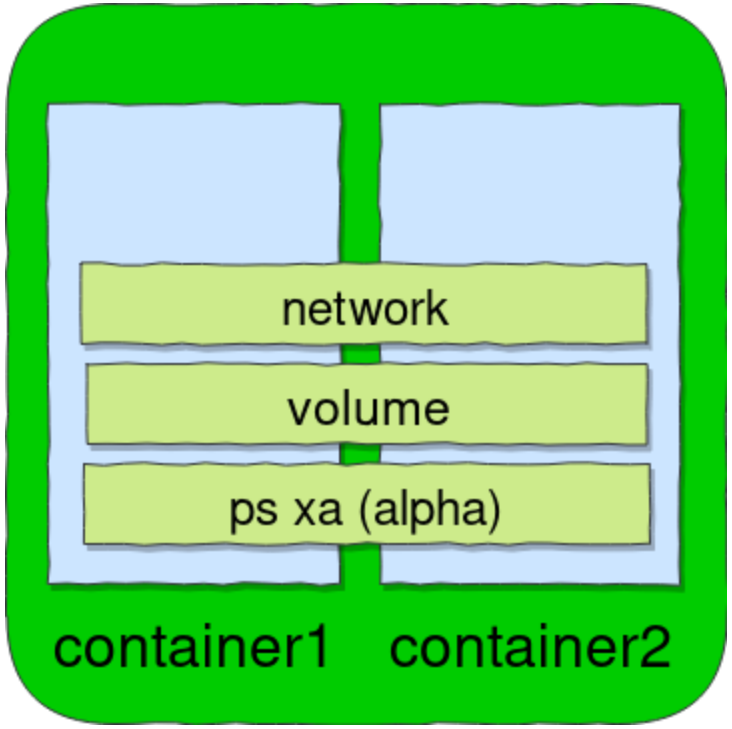

Prometheus
http
Namenode
Datanode


node1
node2
node3
node4
node5



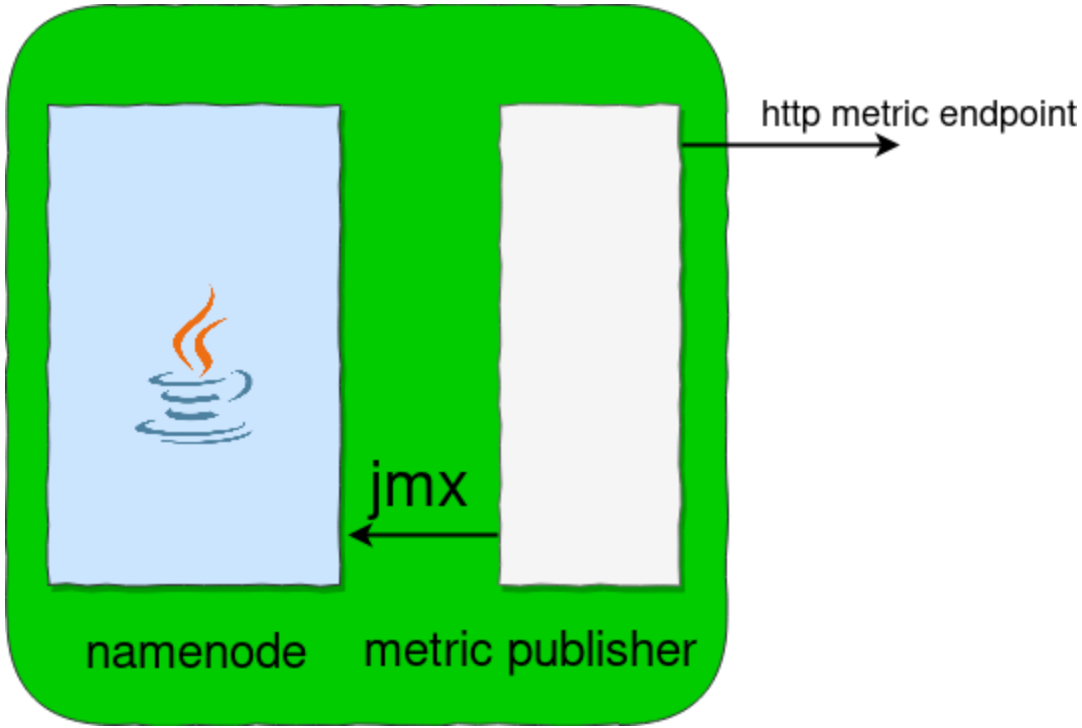

Pod

# Sidecar pattern




Pod

# Sidecar for monitoring over jmx



```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
    spec:
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
```

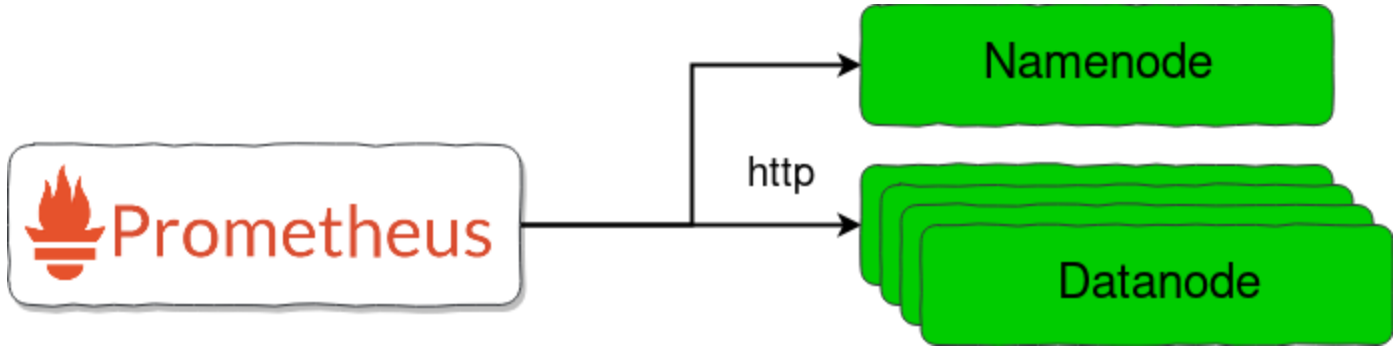
# Flexibility

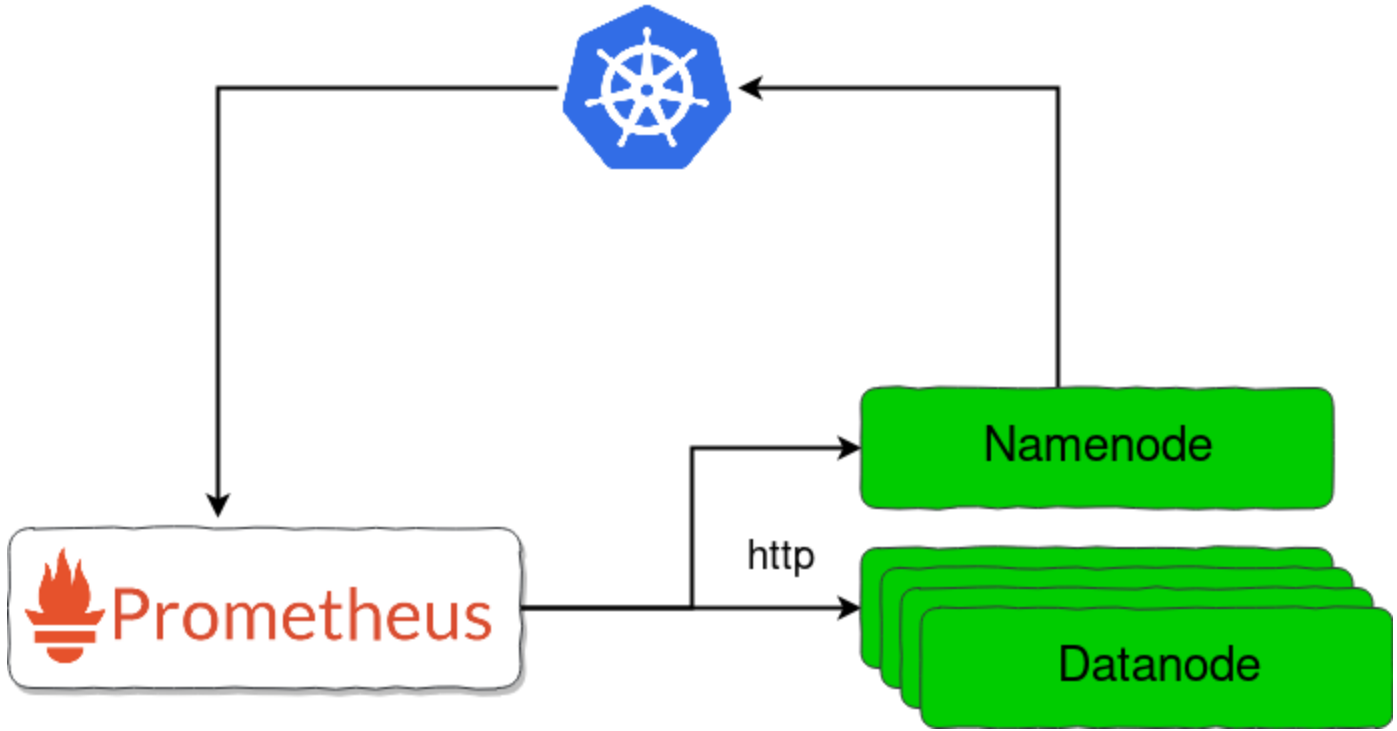
```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
    spec:
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
```

```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
    spec:
      shareProcessNamespace: true
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
        - name: jmxpromo
          image: flokkr/jmxpromo-sidecar
```

# Flexibility

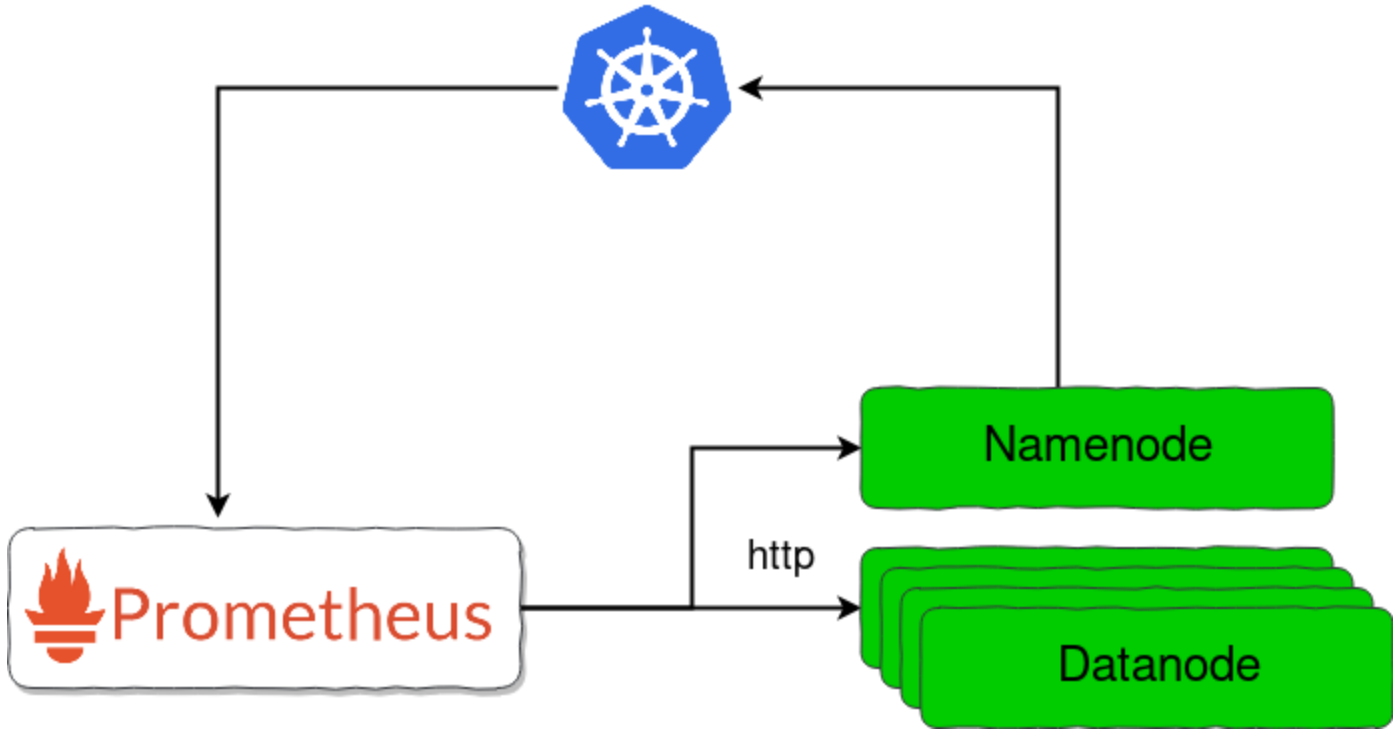
```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
    spec:
      shareProcessNamespace: true
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
        - name: jmxpromo
          image: flokkr/jmxpromo-sidecar
```


Prometheus
http
Namenode
Datanode

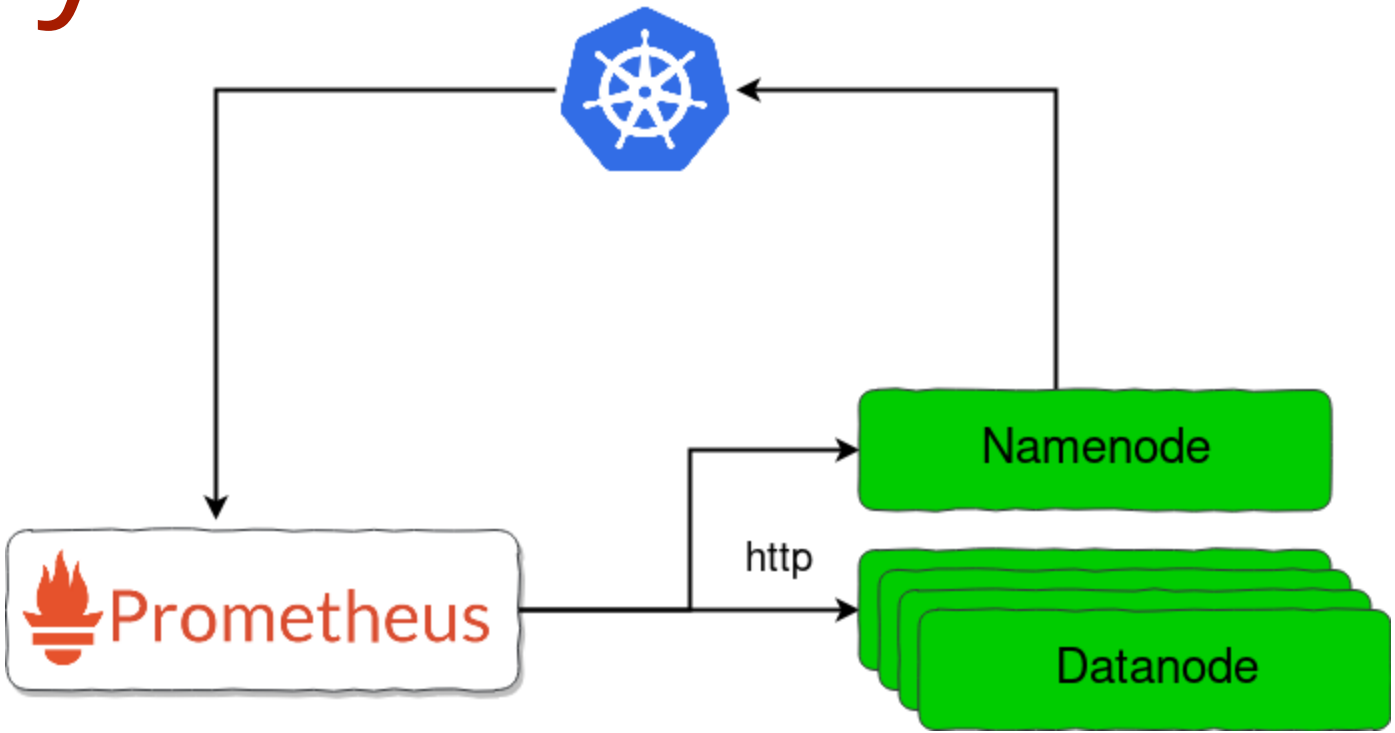

Namenode
http
Prometheus
Datanode

```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
    spec:
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
```

```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "28942"
    spec:
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
```


Namenode
http
Prometheus
Datanode

# Ecosystem




| | |
|---|---|
| **Configuration management** | |
| Source | configmap |
| Preprocessing | helm |
| On change | n/a |
| **Provisioning, Scheduling** | |
| Multihost support | CNI |
| Scheduling | k8s |
| Cluster definition | helm, yaml |
| Scaling | yes |
| Multi tenancy | namespaces |
| Failover | yes |
| **Network** | |
| Intraservice network | CNI |
| DNS | statefulset |
| Service discovery | DNS |
| Data locality | no |
| Availability of the ports | service/ingress |

# Is Hadoop cloud native?

# Is Hadoop cloud native?

Yes, with small modifications

# Is Hadoop cloud native?

Yes, with small modifications

- DNS/IP handling. Shoud work
  - without DNS
  - with changing DNS

# Is Hadoop cloud native?

Yes, with small modifications

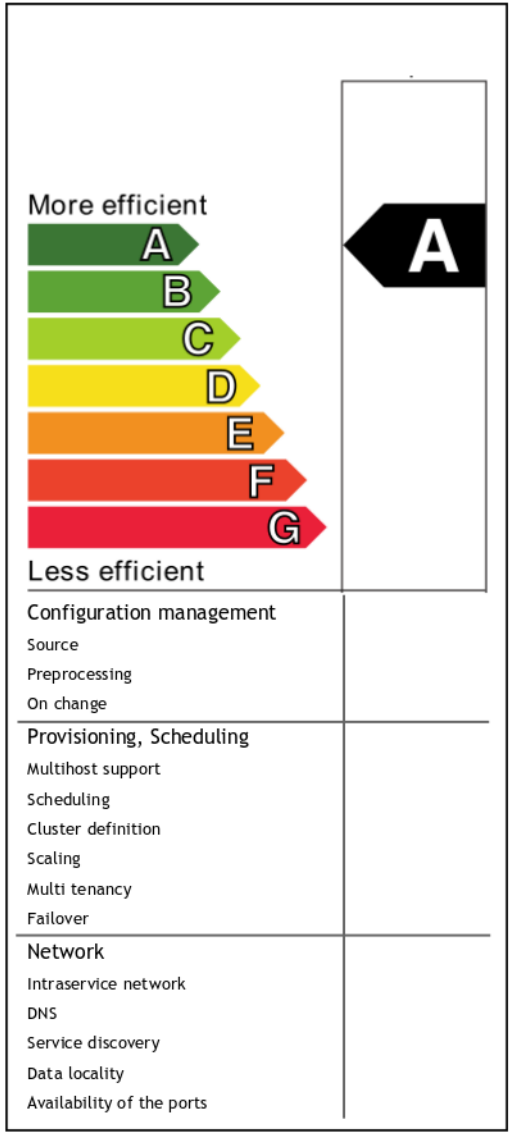
- DNS/IP handling. Shoud work
  - without DNS
  - with changing DNS
- More flexible configuration loading

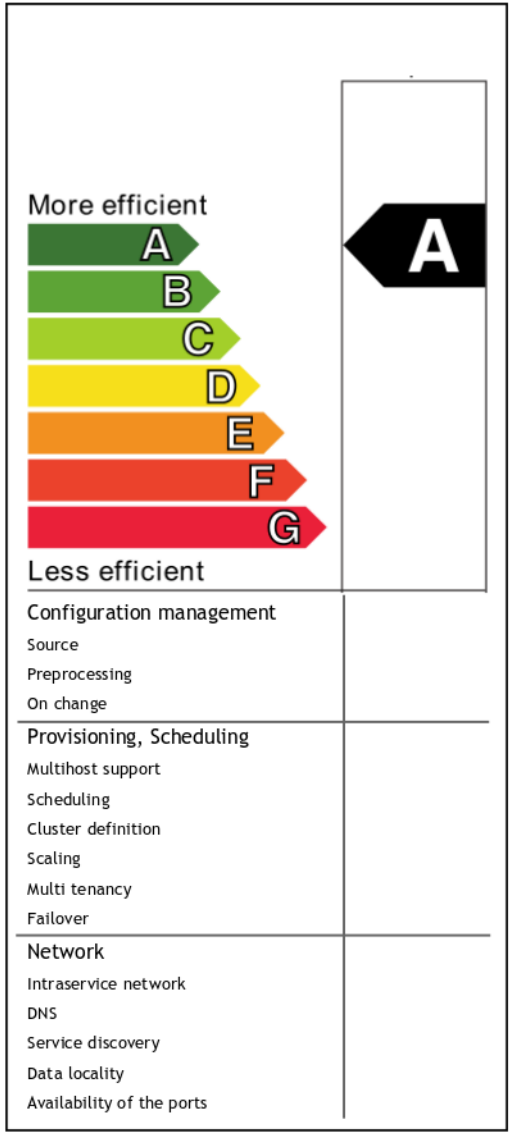
# Is Hadoop cloud native?

Yes, with small modifications

- DNS/IP handling. Shoud work
  - without DNS
  - with changing DNS
- More flexible configuration loading
- Reverse proxy friendly UI

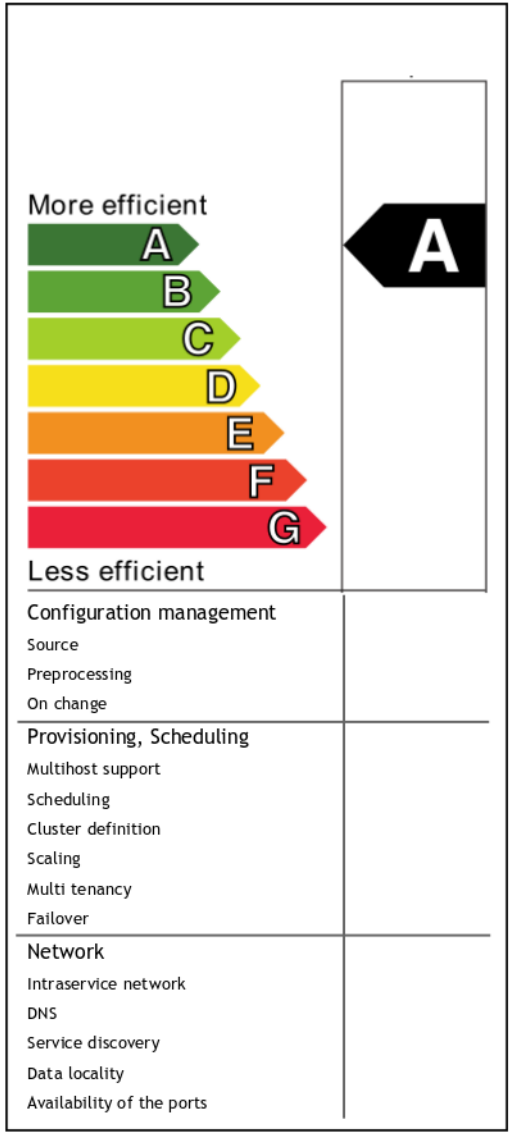
# Summary


More efficient
A
B
C
D
E
F
G
Less efficient
A
Configuration management
Source
Preprocessing
On change
Provisioning, Scheduling
Multihost support
Scheduling
Cluster definition
Scaling
Multi tenancy
Failover
Network
Intraservice network
DNS
Service discovery
Data locality
Availability of the ports




# Summary

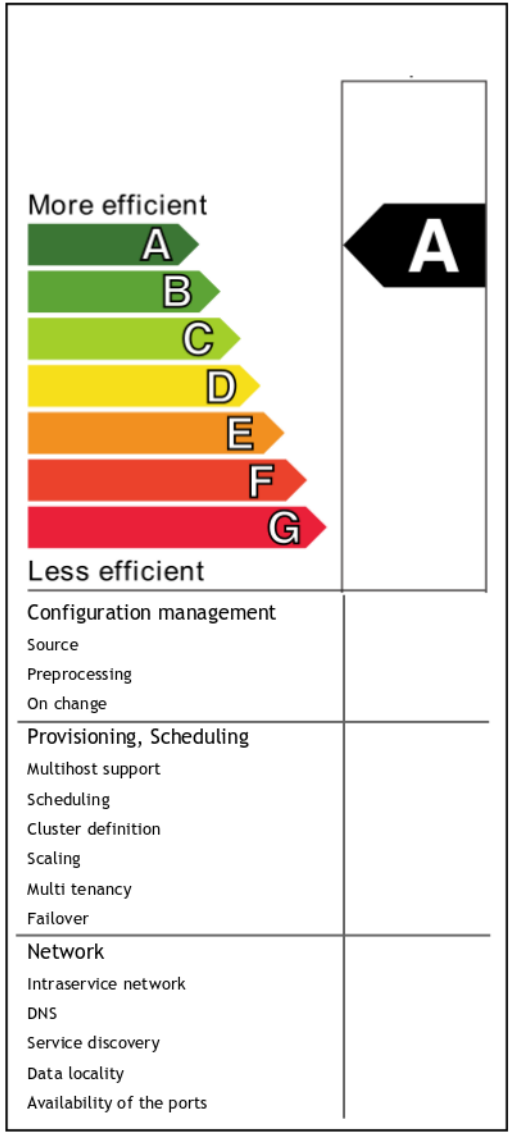
Don't buy without checking the label




# Summary

Don't buy without checking the label

Containerization can help a lot to manage our *Bigdata* clusters




# Summary

Don't buy without checking the label

Containerization can help a lot to manage our *Bigdata* clusters

Hadoop is first class citizen of cloud-native/containerized environments*

# Q&A

**Márton Elek** @anzix

https://flokkr.github.io (bigdata + containers project)

https://github.com/flokkr (source)

elek@apache.org

# Image credits

Yan Pritzker (CC)

Carrie Cizauskas (CC)