From a cluster to the Cloud

Jean-Frederic Clere
Tomcat
@jfclere

APACHECON North America

Sept. 24-27, 2018

Agenda

- Who I am
- A cluster:
  - Session replication and application.
- The cloud:
  - Nope it doesn’t work from scratch.
  - Looking at the different cloud providers.
  - External session replication
  - Modify the tomcat cluster
    - Allow a dynamic list of nodes
    - Only TCP. (8888 port exported via deployment.yml)
  - Demos
- What next? Questions / Suggestions

Who am I?

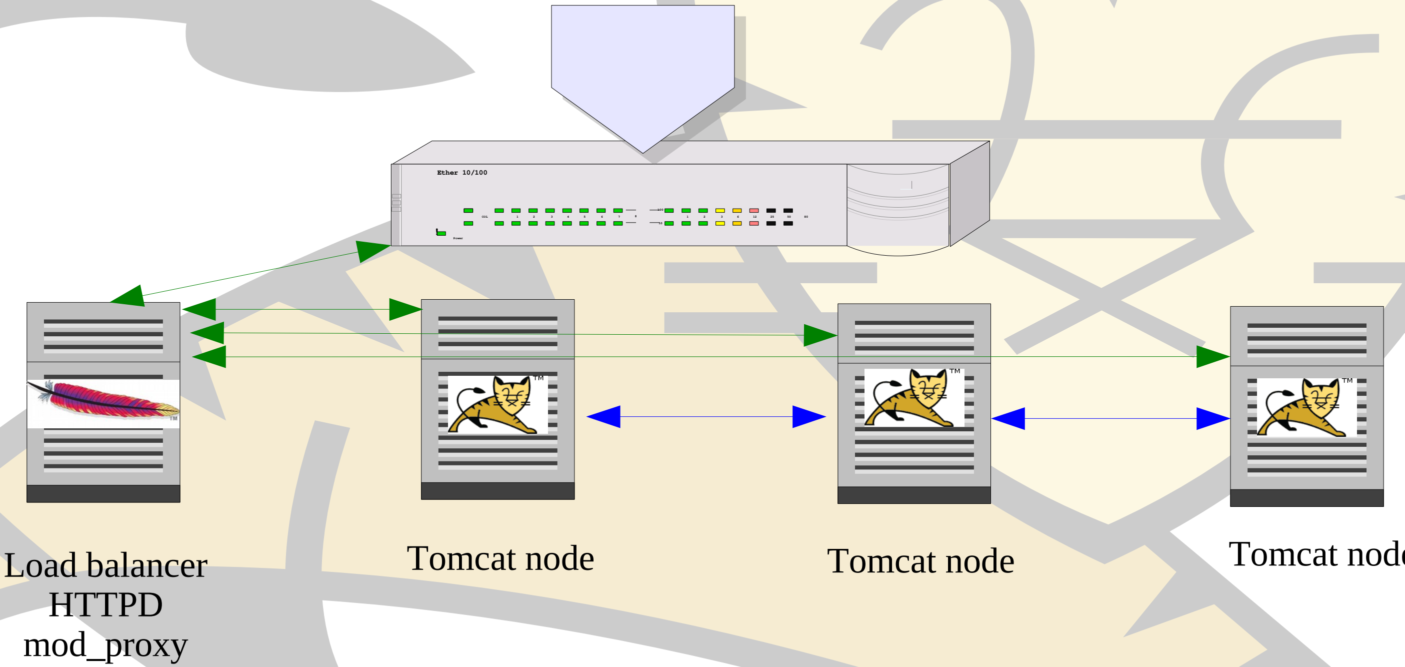
Jean-Frederic Clere

- Red Hat
- Years writing JAVA code and server software
- Tomcat committer since 2001
- Doing OpenSource since 1999
- Cyclist/Runner etc
- Lived 15 years in Spain (Barcelona)
- Now in Neuchâtel (CH)

Session replication in a cluster

- HTTP/1.1
  - No transaction
  - No persistent connection
- Web App:
  - Using cookies to carry session ID
  - Store information in the session:
    - Shopping cart etc.
- Multi nodes and dynamic
  - Route request to right node
  - Replicate information

A cluster


Ether 10/100
Load balancer
HTTPD
mod_proxy
Tomcat node
Tomcat node
Tomcat node

How to replicate sessions

In cluster:

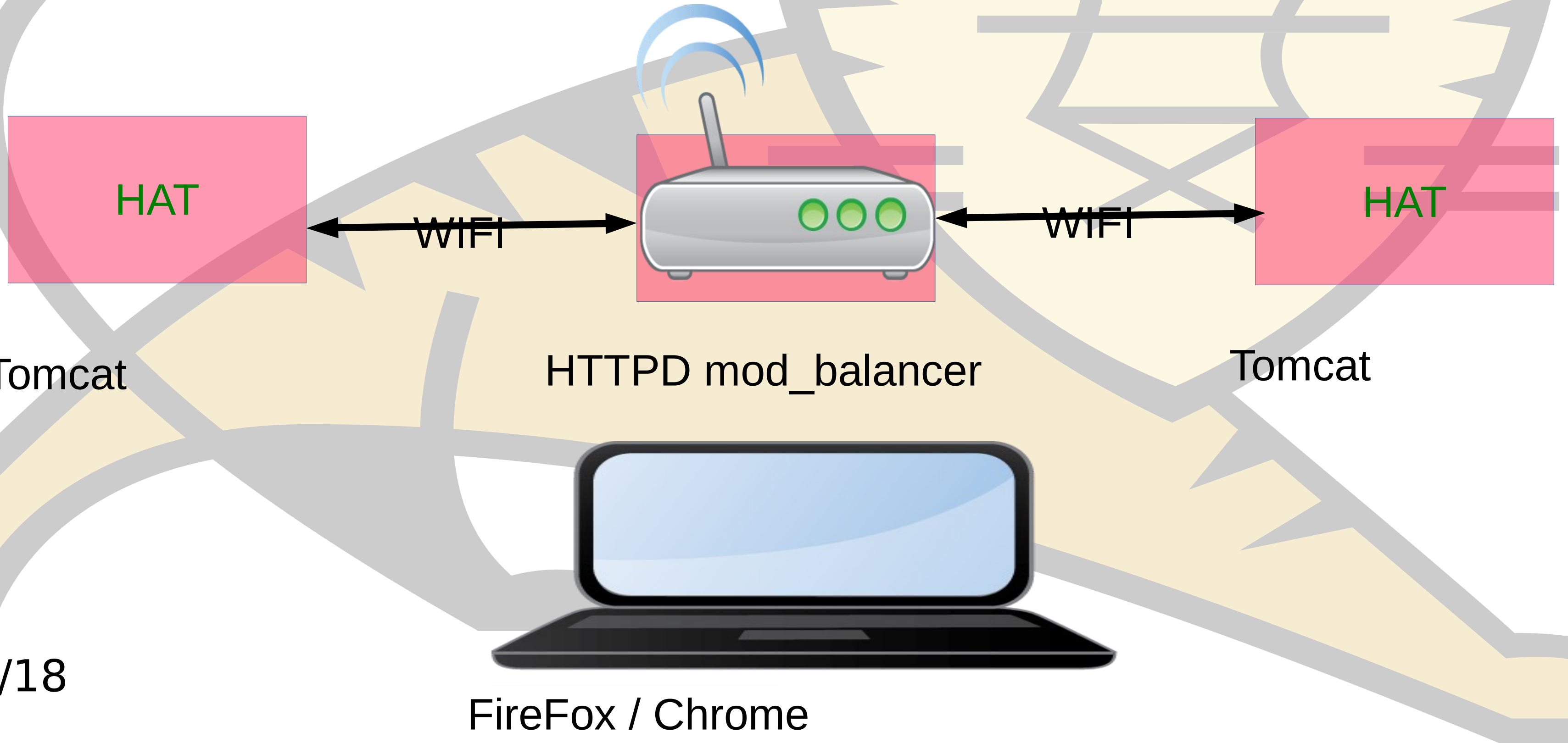
<distributable/> in web.xml

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>

Port upd 45564

Ports tcp range 4000:4100

Cluster Demo



Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. *https://kubernetes.io/*

Cloud providers

Most of the major cloud providers rely on Kubernetes as a container management solution.

amazon web services

IBM

Google Cloud Platform

Azure

Cloud providers

We worked on adding support for Kubernetes so that our solution would be available on all of these providers.

amazon web services ✓

IBM

Google Cloud Platform ✓

Azure ✓

Deployment

Build

Image of tomcat-in-the-cloud

Push

Docker registry

Run

Docker container

Expose

Port 80

Automation

Because the deployment can be time consuming and slightly different for each of the cloud providers (in terms of permission management). We’re currently working on automating the process.

AWS:

- **awscli /IAM console / docker / kops / kubectl**

Azure:

- **azure-cli /Azure console / docker / kubectl**

Google:

- **google-cloud-sdk / google cloud console / docker / kubectl**

OPENSHIFT

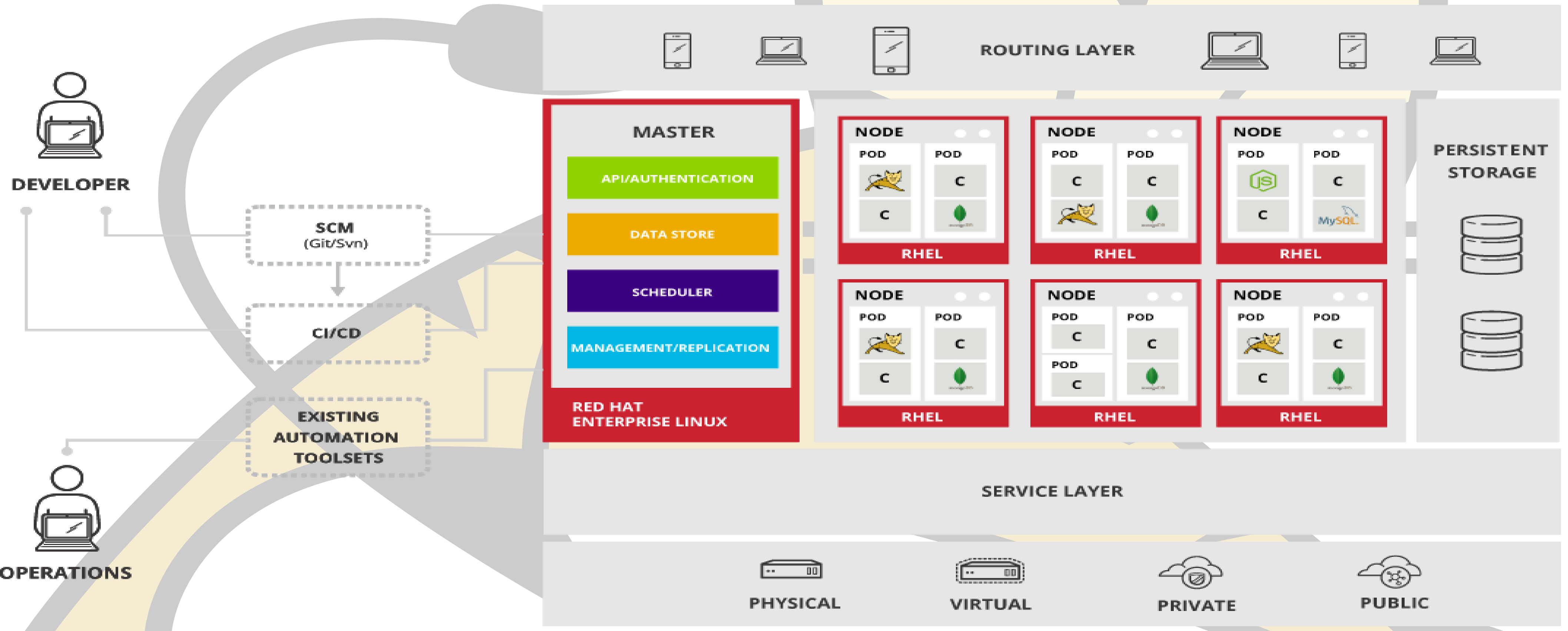
A Red Hat project / product

See OpenShift

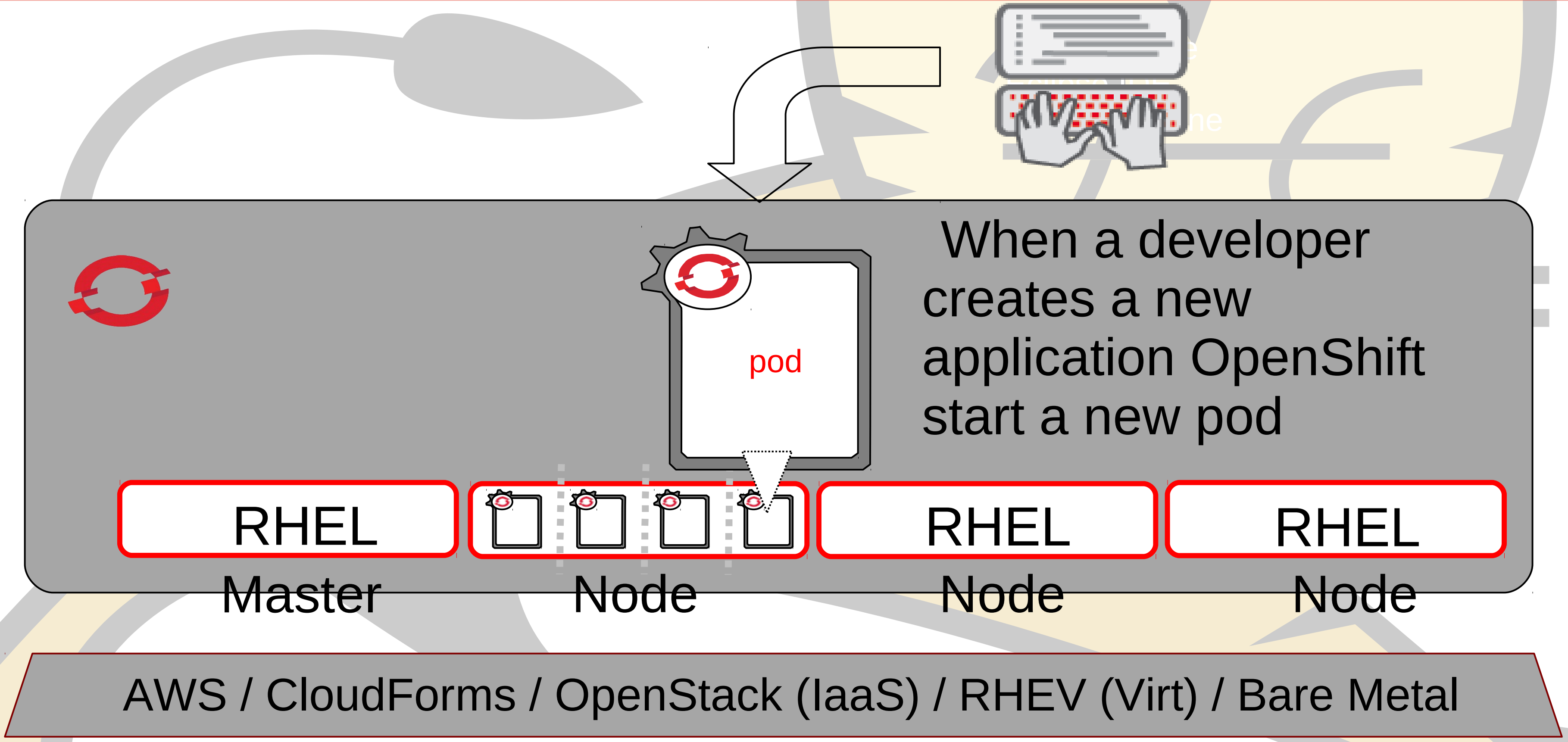
https://www.openshift.com/

Can use AWS (public cloud) or Private on premise.

Tomcat in OpenShift/Kubernetes



Developping Tomcat App in OpenShift/Kubernetes


pod
RHEL
Master
Node
RHEL
Node
RHEL
Node
AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

When a developer creates a new application OpenShift start a new pod

Getting started

minishift:

- Allows a demo on a single box.
- Easy to setup
- Small demo

Online:

- We have prepared wiki to help you to start:
  - https://github.com/web-servers/tomcat-in-the-cloud/wiki
- We have a katacoda tutorial:
  - https://katacoda.com/jfclere/courses/tomcat-in-the-cloud

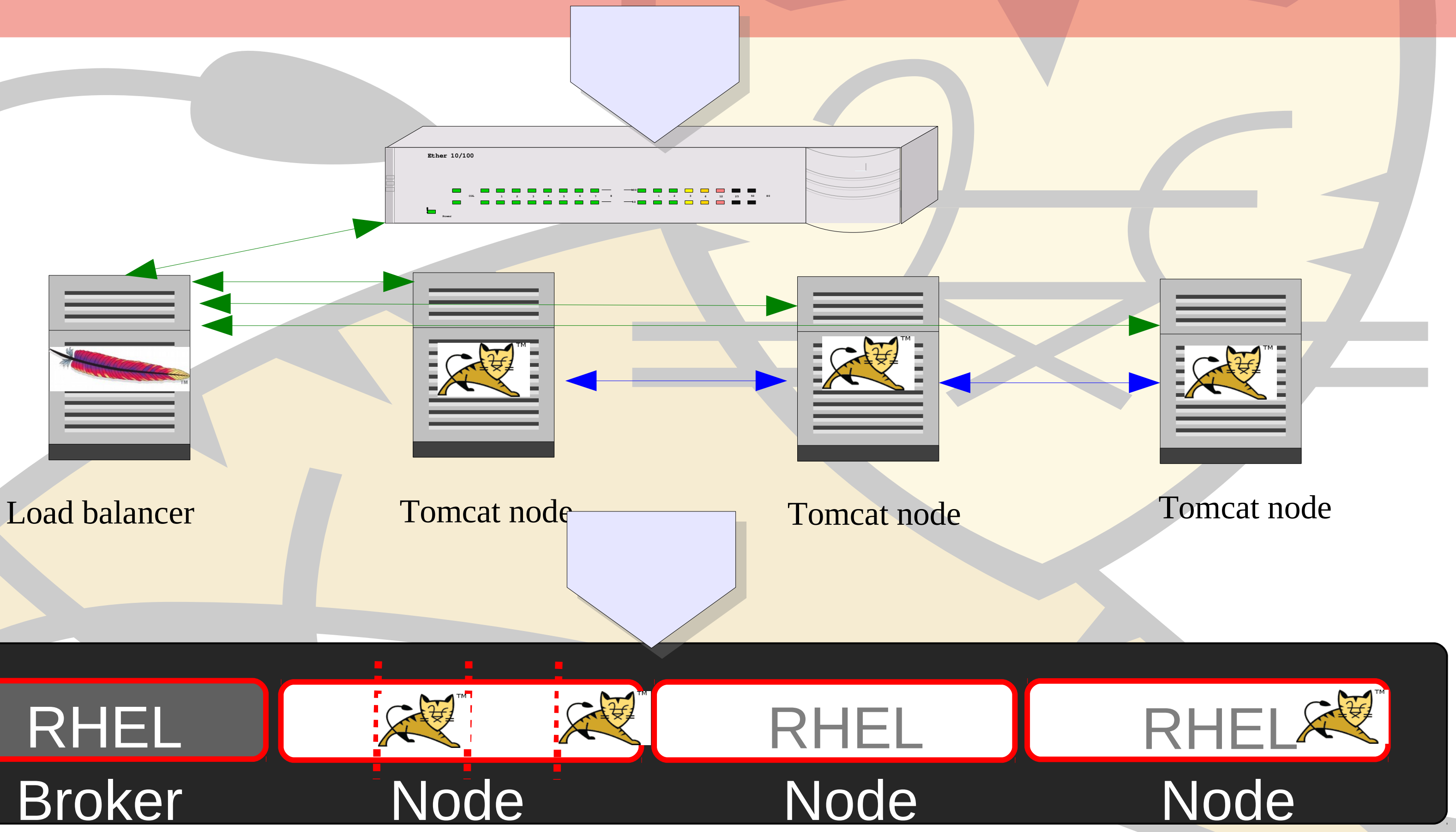
Bare metal / VM:

- Use ansible to install
- 2 nodes + master minimal

Tomcat webapp with sessions

- Rest Counter demo.

From a cluster to the Cloud

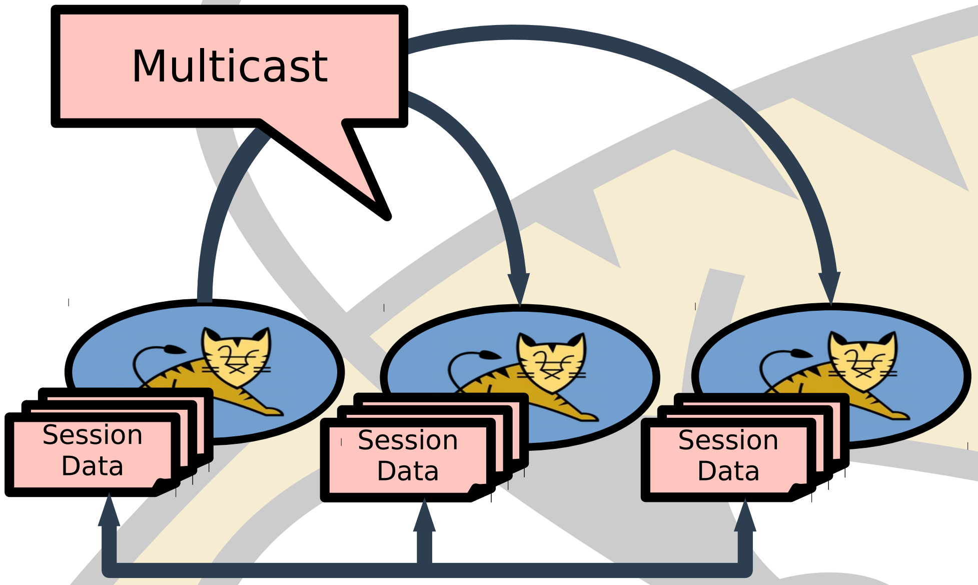


Problems for a cluster to cloud...

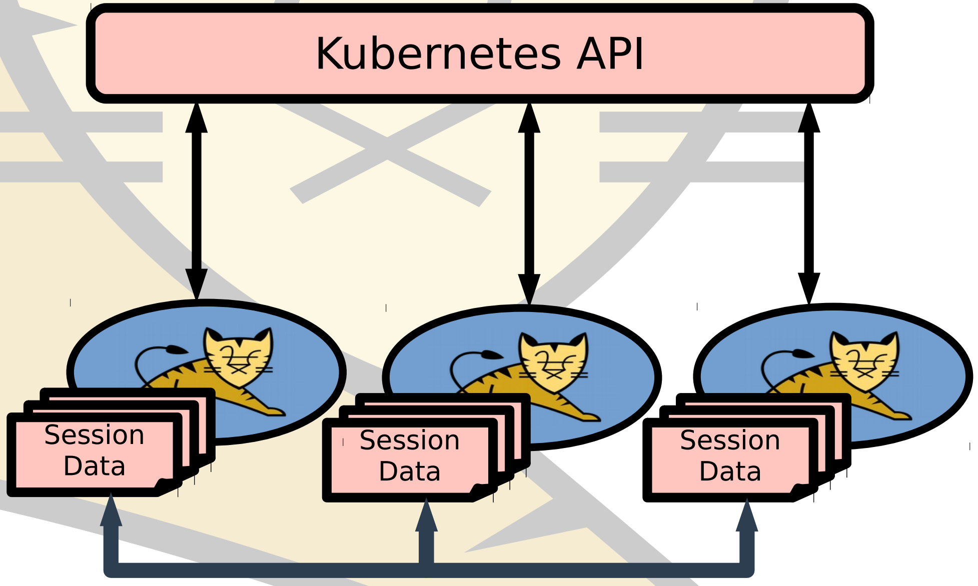
- Many ways to solve:
  - Embed tomcat with SpringBoot
  - Create a docker image
  - Extend an existing docker image
  - Fabric8
- Tomcat session replication:
  - No multicast in the cloud.
  - Need a “ping” to find the other nodes (KubePing)
  - Need to add “view nodes” permission to the system account of the project.

Solutions

Tomcat cluster built-in solution
Peer discovery through multicast heartbeat messages
Does not work in a cloud environment



Our solution
Peer discovery through Kubernetes Downward API
Works in all kubernetes clouds



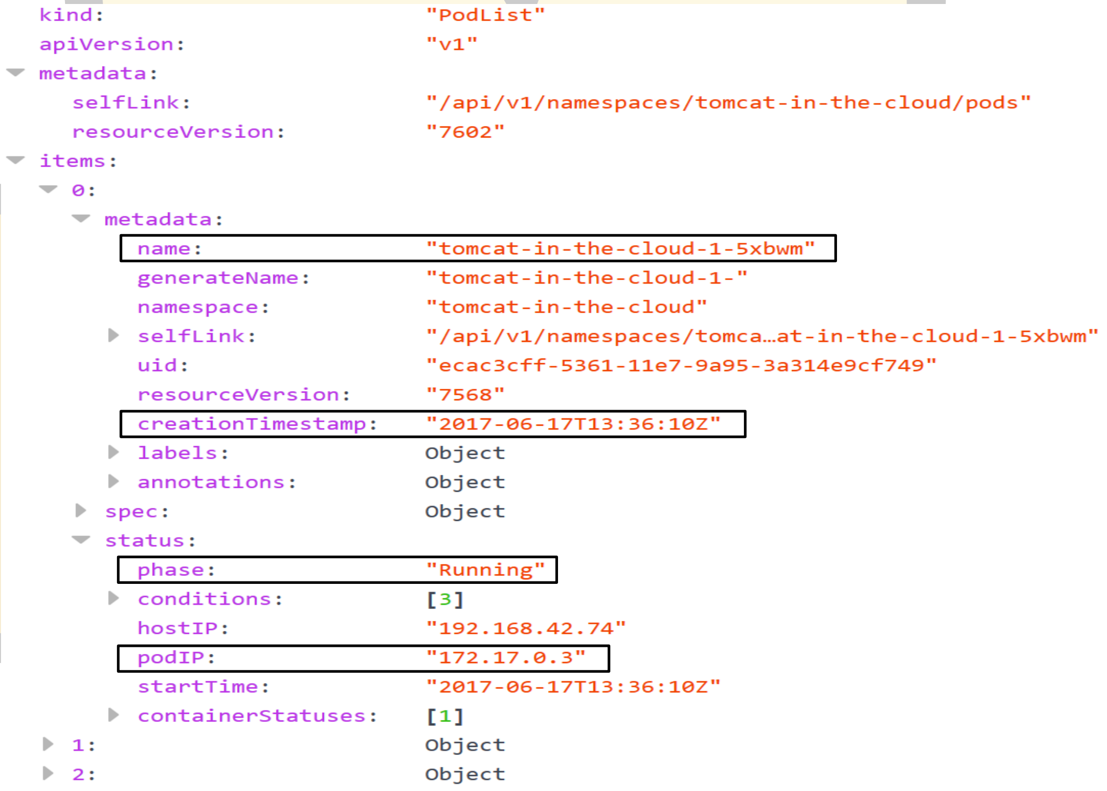
Kubernetes API

Tools for managing a Kubernetes cluster

Accessible from the pods within the cluster

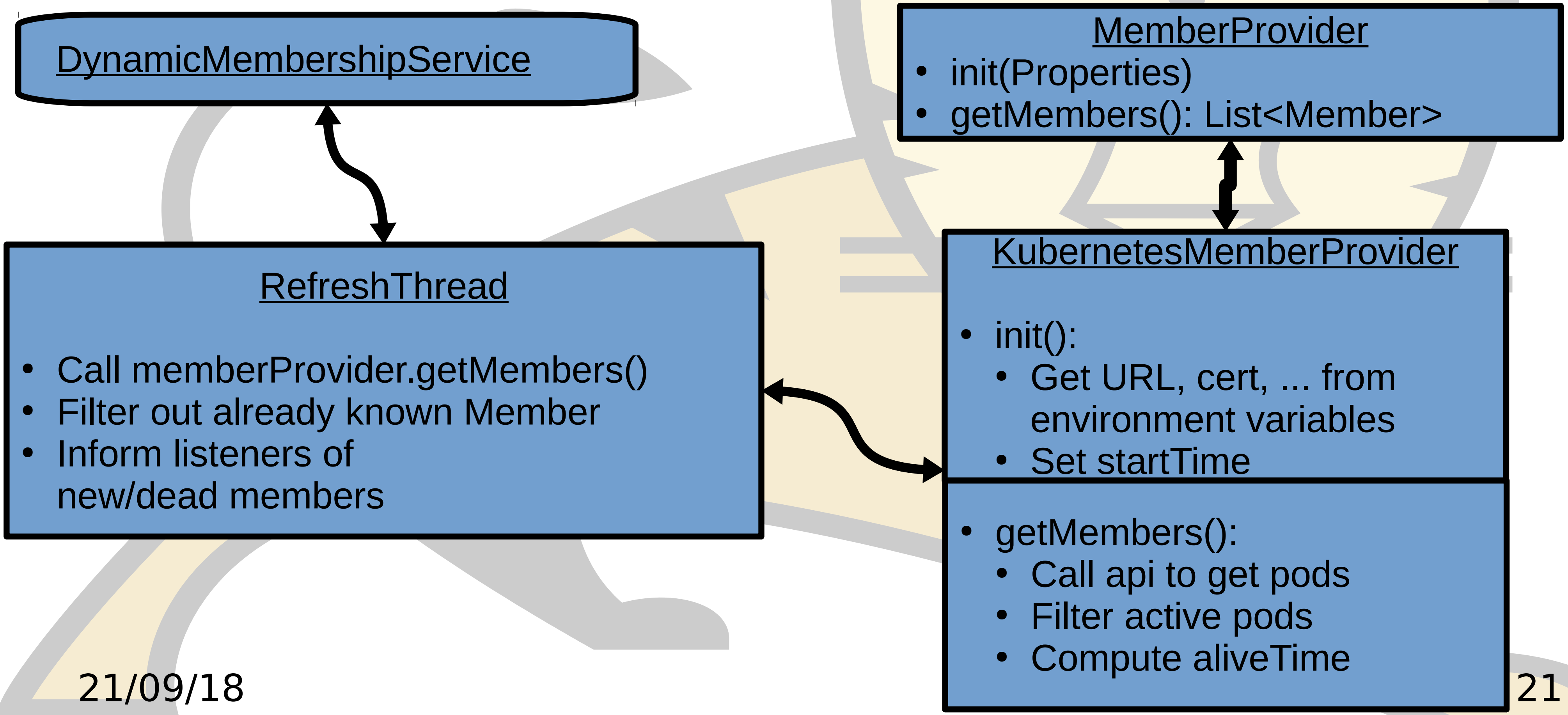
GET /api/v1/namespaces/tomcat-in-the-cloud/pods

→ Return a JSON representation of all the pods in the cluster



```
kind:                        "PodList"
apiVersion:                  "v1"
▼ metadata:
    selfLink:                "/api/v1/namespaces/tomcat-in-the-cloud/pods"
    resourceVersion:         "7602"
▼ items:
  ▼ 0:
    ▼ metadata:
        name:                "tomcat-in-the-cloud-1-5xbwm"
        generateName:        "tomcat-in-the-cloud-1-"
        namespace:           "tomcat-in-the-cloud"
      ▶ selfLink:            "/api/v1/namespaces/tomca…at-in-the-cloud-1-5xbwm"
        uid:                 "ecac3cff-5361-11e7-9a95-3a314e9cf749"
        resourceVersion:     "7568"
        creationTimestamp:   "2017-06-17T13:36:10Z"
      ▶ labels:              Object
      ▶ annotations:         Object
    ▶ spec:                  Object
    ▼ status:
        phase:               "Running"
      ▶ conditions:          [3]
        hostIP:              "192.168.42.74"
        podIP:               "172.17.0.3"
        startTime:           "2017-06-17T13:36:10Z"
      ▶ containerStatuses:   [1]
  ▶ 1:                       Object
  ▶ 2:                       Object
```

Architecture



- init(Properties)
- getMembers(): List<Member>

- Call memberProvider.getMembers()
- Filter out already known Member
- Inform listeners of new/dead members

- init():
  - Get URL, cert, ... from environment variables
  - Set startTime
- getMembers():
  - Call api to get pods
  - Filter active pods
  - Compute aliveTime

What is done

Demo contents:

- **Embedded Tomcat**
- **HypriotOS + Fedora with Oracle JVM (for RPI3 demo)**
- **Reuse existing tcp cluster code**

Some code still missing:

- **Some in Tomcat (one PR missing)**
- **Documentation / tests.**

Some more stuff:

- **We use ansible for the install.**
- **Some maven builds and shells.**

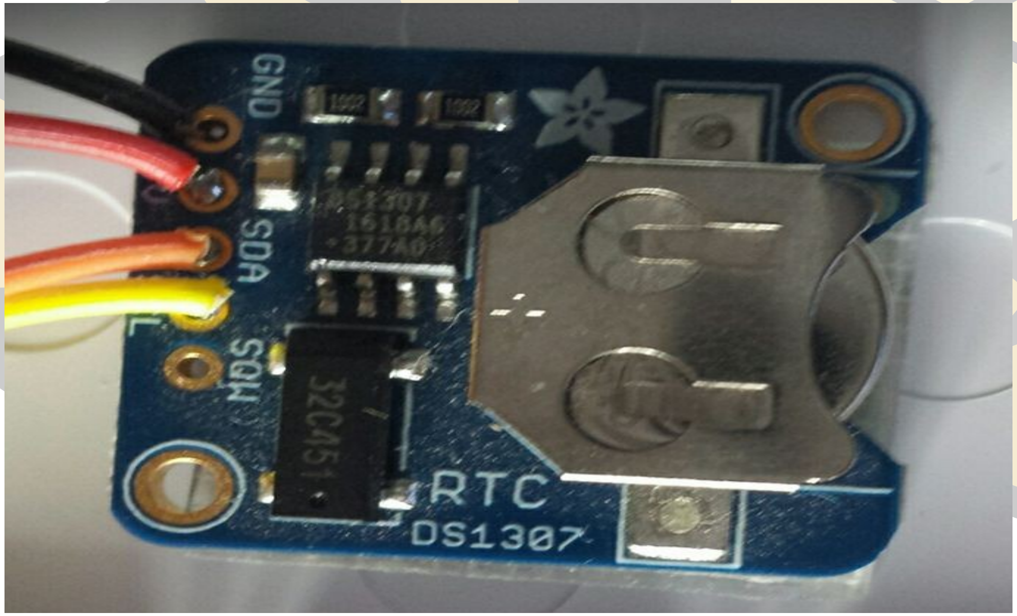
What to do “step by step”

Make sure you have hard clock when no Internet.

I use a Timer Server in the captive portal RPI3.

Chronyd (NTP when on line & RTC otherwise).

See My blog on ds1307-on-rpi3



What to do next for each node of the on premise cloud you are building

Install HypriotOS on the 3 nodes

Download the image from Hypriot downloads page.

Extract and use dd to copy the image on the sd card

Boot the RPI3 with the image

Connect the RPI3 to an Ethernet port of your router

Get the IP for RPI3 using nmap

```
sudo nmap -sn 192.168.1.0/24
```

```
Nmap scan report for pc-8.home (192.168.1.108)
Host is up (0.087s latency).
MAC Address: B8:27:EB:7A:A6:98 (Raspberry Pi Foundation)
```

Configure each node to use WIFI (easier than cables)

- **Make sure the captive portal is working and does Nat (and is connected)**
- **In the node add in /etc/network/interfaces.d/wlan0**

```
auto wlan0
iface wlan0 inet dhcp
  wireless-essid PI
  wireless-mode Managed
```

- **Use ifup wlan0 to start the WIFI**
- **Check you can ping the internet and download stuff when installing.**

Use ansible to install kubernetes on each node

Clone ansible project to install kubernetes on Hypriot

Create your ansible list of nodes like

```
[pis]
10.0.0.204 name=n0 host_extra="master registry"
10.0.0.203 name=n1
10.0.0.202 name=n2

[master]
10.0.0.204

[nodes]
10.0.0.202
10.0.0.203
```

Start the installation (you might fill .ssh/authorized_keys before)

```
ansible-playbook -k -i hosts setup.yml
```

Check that everything is working

export KUBECONFIG=./run/pi-cluster.cfg

kubectl get nodes

```
NAME    STATUS    AGE
n0      Ready     77d
n1      Ready     77d
n2      Ready     77d
```

Preparing the docker image

- **Build the uber jar (mvn install in tomcat-in-the-cloud)**
- **With docker on any of the nodes**
- **Create the image based on https://github.com/fabric8io-images/java/**

```
docker build .
docker images
docker run -i -t 4a1b89814050
docker tag 4a1b89814050 jfclere/armv7fabric8:1.0.0
```

- **push it with a tag:**

```
docker push jfclere/armv7fabric8:1.0.0
```

- **https://hub.docker.com/r/jfclere/armv7fabric8/**

Creating the user and role in kubernetes for the kubeping

Create the system account

```
kubectl create -f serviceaccount.yaml
```

Create a role to get, watch and list the pods of our namespace

```
kubectl create -f role.yaml
```

Create the user

```
kubectl create -f user.yaml
```

Create our pods using the docker image

```
kubectl create -f tomcat-rpi3.json
```

https://github.com/jfclere/tomcatPI/blob/master/cloud/role.yaml

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
 verbs: ["get", "watch", "list"]
```

https://github.com/jfclere/tomcatPI/blob/master/cloud/user.yaml

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: system:serviceaccount:default:tomcat-in-the-cloud
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

https://github.com/jfclere/tomcatPI/blob/master/cloud/serviceaccount.yaml

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ServiceAccount
metadata:
name: tomcat-in-the-cloud
```

https://github.com/jfclere/tomcatPI/blob/master/cloud/tomcat-rpi3.json

```
"apiVersion": "apps/v1beta1",
"kind": "Deployment",
"metadata": {
 "name": "tomcat-in-the-cloud"
},
"spec": {
 "replicas": 2,
 "template": {
  "metadata": {
   "labels": {
    "app": "tomcat-in-the-cloud"
   }
  },
  "spec" : {
   "serviceAccountName": "tomcat-in-the-cloud",
   "serviceAccount": "tomcat-in-the-cloud",
   "containers": [
    {
     "name": "tomcat-in-the-cloud",
     "image": "jfclere/armv7fabric8:1.0.0",
```

https://github.com/jfclere/tomcatPI/blob/master/cloud/tomcat-rpi3.json

```
"name": "tomcat-in-the-cloud",
"image": "jfclere/armv7fabric8",
"ports": [
  {
   "containerPort": 8080
  }
],
"env": [
  {
   "name": "OPENSHIFT_KUBE_PING_NAMESPACE",
   "value": "default"
  },
  {
   "name": "JAVA_APP_JAR",
   "value": "tomcat-in-the-cloud-1.0-SNAPSHOT.jar"
  },
  {
   "name": "KUBERNETES_RO_SERVICE_HOST",
   "value": "127.0.0.1"
  },
  {
   "name": "KUBERNETES_RO_SERVICE_PORT",
   "value": "8001"
```

Make the application accessible

Expose deployment

```
kubectl expose deployment tomcat-in-the-cloud --type=NodePort --name=tomcat-in-the-cloud
```

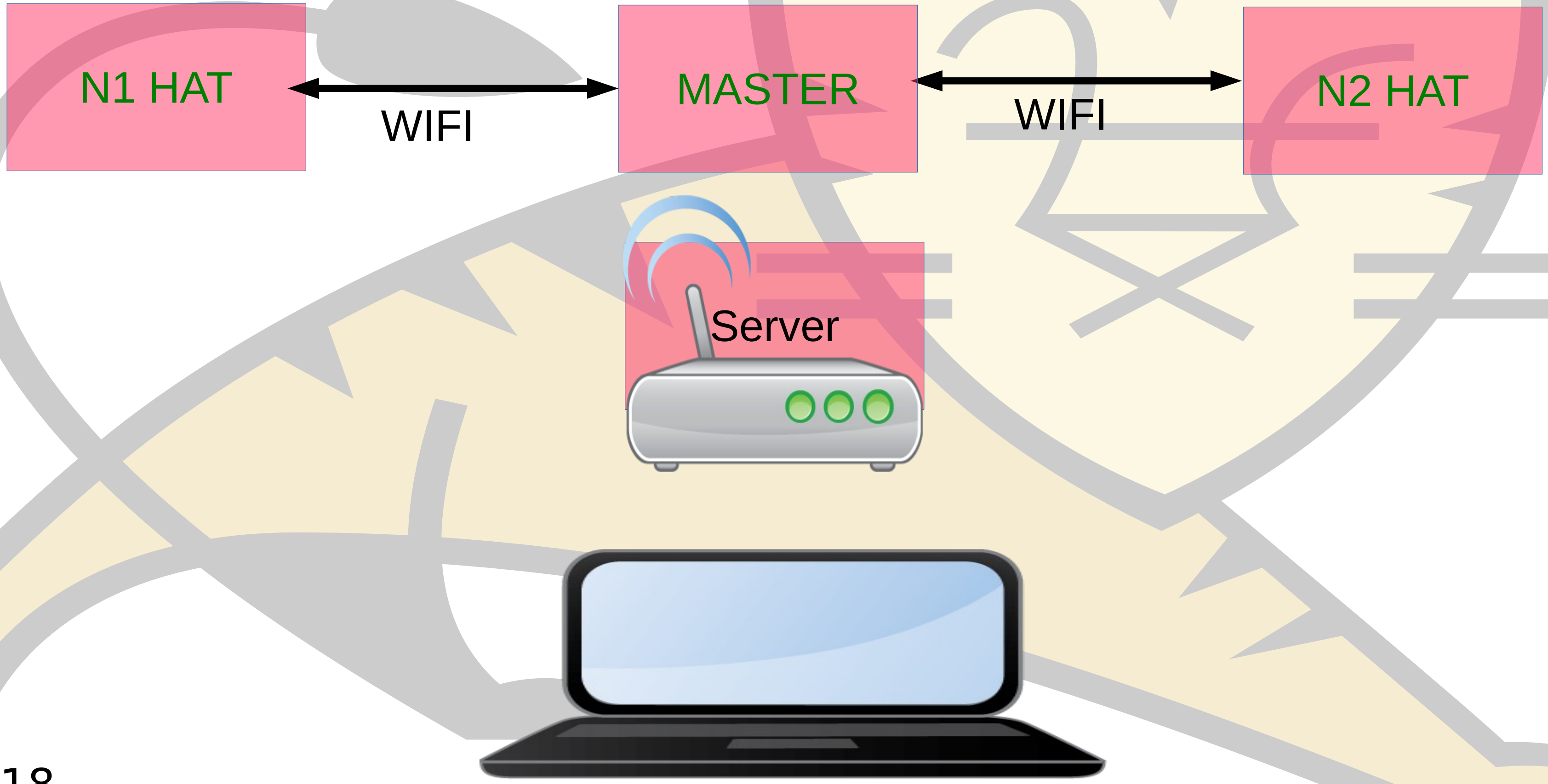
Read the node port (RPI3) / (ip or hostname for online clouds)

```
kubectl describe services tomcat-in-the-cloud
NodePort <unset>   32206/TCP
```

Test it curl for example

```
curl -v --cookie "JSESSIONID=4833B5E258B2022A600851E9AB29B8FA" http://10.0.0.204:32206/
{
  "counter": 4,
  "id": "4833B5E258B2022A600851E9AB29B8FA",
  "new": false,
  "server": "10.40.0.2",
  "hostname": "tomcat-in-the-cloud-3133120499-bks16"
}
```

Bare Metal Cloud demo



FireFox / Chrome / ansible

Katacoda tutorial

https://katacoda.com/jfclere/courses/tomcat-in-the-cloud

And the sources:

https://github.com/jfclere/intro-katacoda/tree/master/tomcat-in-the-cloud/deploy-titc-using-cli

That is just what you have to do if you have a cloud ready to use…

Where we are

Main sites:

- https://github.com/web-servers/tomcat-in-the-cloud
- https://github.com/jfclere/tomcatPI
- https://docs.openshift.com
- https://github.com/Project31

Thanks:

- Université de Neuchâtel
- Kurt Stam <kstam@redhat.com>

Questions ?
Suggestions?

THANK YOU

YOUR NAME

@jclere

jfclere@gmail.com