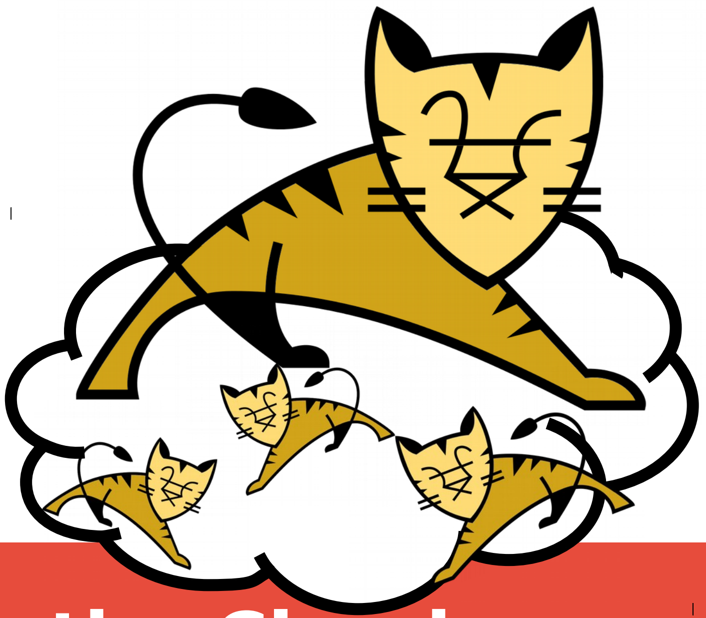
R&D Workshop @ Red Hat



Tomcat Session Replication in the Cloud

Midpoint presentation

Supervisor: Jean-Frederic Clere

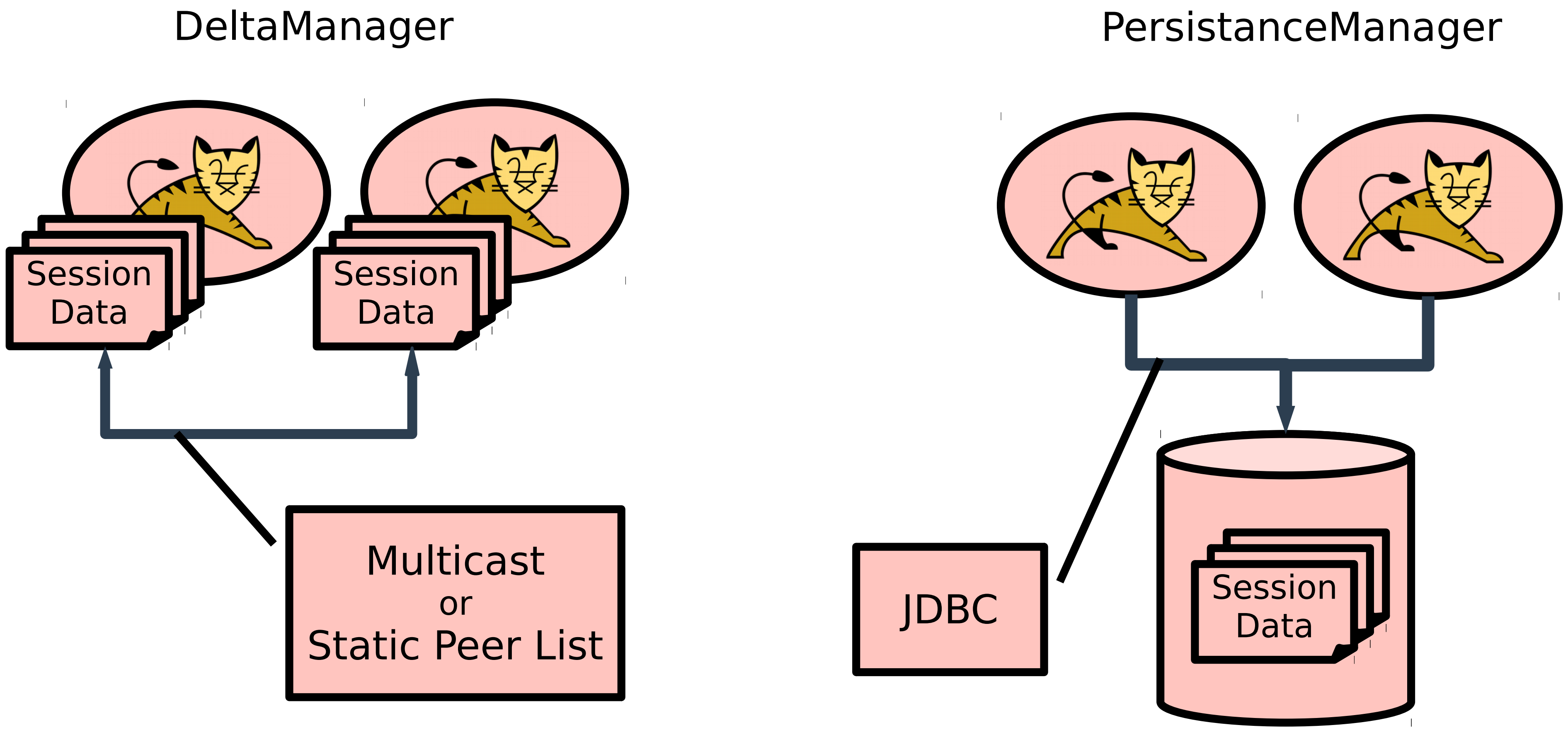
Ismaïl Senhaji, Guillaume Pythoud

Outline

- Reminder
- What has been done
- What's next
- Changes to the plan

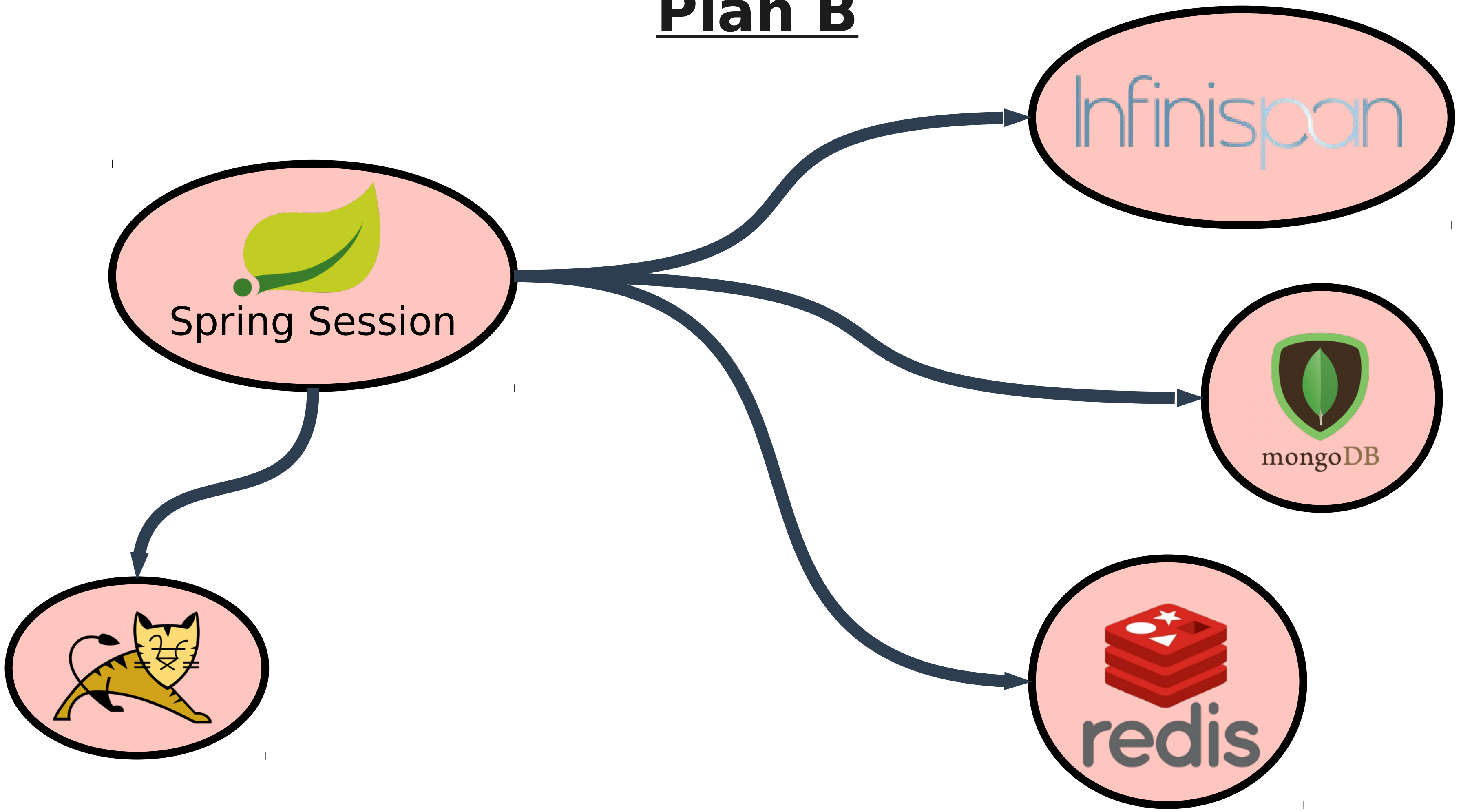
Reminder

Plan A



Reminder

Plan B



Reminder

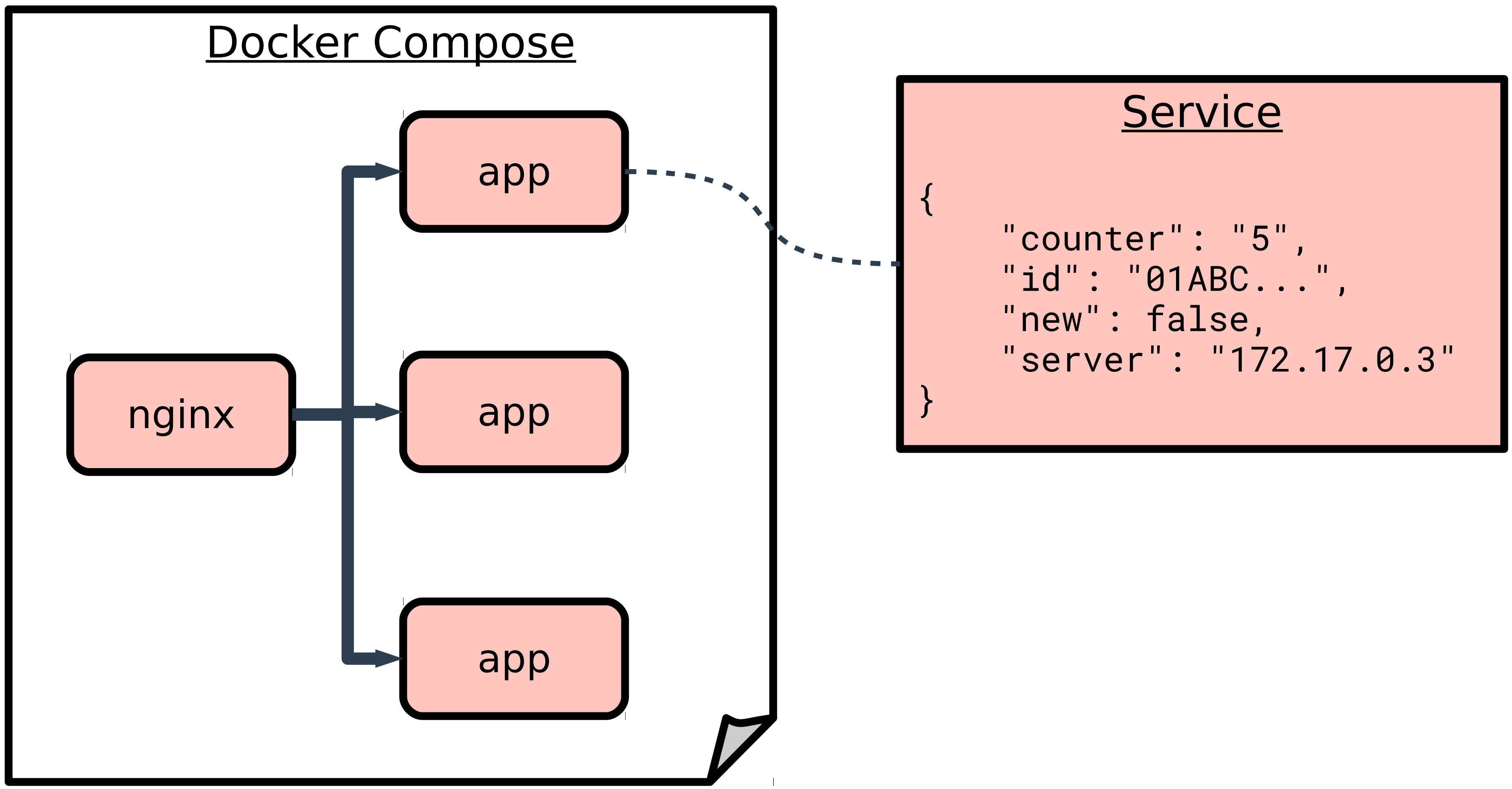
Pros and cons:

	DeltaManager	PersistanceManager	Spring Session
+	• Built-in → no dependencies	• Works	• Works • Well-documented
-	• Multicast • Difficult	• Must run DB server • Bad performance	• Dependecy on Spring • Must run DB server

What has been done

- Many new technologies
  - OpenShift
  - Kubernetes
  - Docker
  - Fabric8
  - Tomcat
- Best way to learn is to experiment
- We tried the tools out, got familiar with them

Experiment setup



```
{
    "counter": "5",
    "id": "01ABC...",
    "new": false,
    "server": "172.17.0.3"
}
```

Experiment 1

Tomcat + JSP

- Multicast peer discovery
- Configuration with *server.xml* file
- Just one line to add:

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

Experiment 2

Spring Boot

- No *server.xml* file for configuration
  ➔ Embedded Tomcat must be configured programmatically
- Spring Boot developers didn’t have our use-case in mind
  ➔ Messy workaround

```java
@Configuration
public class TomcatConfiguration
{
    @Bean
    public EmbeddedServletContainerFactory servletContainerFactory()
    {
        TomcatEmbeddedServletContainerFactory factory = new TomcatEmbeddedServletContainerFactory()
        {
            @Override
            protected TomcatEmbeddedServletContainer getTomcatEmbeddedServletContainer(
                    Tomcat tomcat)
            {
                configureCluster(tomcat);
                return super.getTomcatEmbeddedServletContainer(tomcat);
            }

            private void configureCluster(Tomcat tomcat)
            {
                // static membership cluster

                SimpleTcpCluster cluster = new SimpleTcpCluster();
                cluster.setChannelStartOptions(3);
                {
                    DeltaManager manager = new DeltaManager();
                    manager.setNotifyListenersOnReplication(true);
                    cluster.setManagerTemplate(manager);
                }
                {
                    GroupChannel channel = new GroupChannel();
                    {
                        NioReceiver receiver = new NioReceiver();
                        receiver.setPort(localClusterMemberPort);
                        channel.setChannelReceiver(receiver);
                    }
                    {
                        ReplicationTransmitter sender = new ReplicationTransmitter();
                        sender.setTransport(new PooledParallelSender());
                        channel.setChannelSender(sender);
                    }
                    channel.addInterceptor(new TcpPingInterceptor());
                    channel.addInterceptor(new TcpFailureDetector());
                    channel.addInterceptor(new MessageDispatchInterceptor());
                    {
                        StaticMembershipInterceptor membership =
                                new StaticMembershipInterceptor();
                        String[] memberSpecs = clusterMembers.split(",", -1);
                        for (String spec : memberSpecs)
                        {
                            ClusterMemberDesc memberDesc = new ClusterMemberDesc(spec);
                            StaticMember member = new StaticMember();
                            member.setHost(memberDesc.address);
                            member.setPort(memberDesc.port);
                            member.setDomain("MyWebAppDomain");
                            member.setUniqueId(memberDesc.uniqueId);
                            membership.addStaticMember(member);
                        }
                        channel.addInterceptor(membership);
                    }
                    cluster.setChannel(channel);
                }
                cluster.addValve(new ReplicationValve());
                cluster.addValve(new JvmRouteBinderValve());
                cluster.addClusterListener(new ClusterSessionListener());

                tomcat.getEngine().setCluster(cluster);
            }

        };

        factory.addContextCustomizers(new TomcatContextCustomizer()
        {
            @Override
            public void customize(Context context)
            {
                context.setManager(new DeltaManager());
                context.setDistributable(true);
            }
        });

        return factory;
    }

    private static class ClusterMemberDesc
    {
        public String address;
        public int port;
        public String uniqueId;

        public ClusterMemberDesc(String spec) throws IllegalArgumentException
        {
            String[] values = spec.split(":", -1);
            if (values.length != 3)
                throw new IllegalArgumentException("clusterMembers element " +
                        "format must be address:port:uniqueIndex");
            address = values[0];
            port = Integer.parseInt(values[1]);
            int index = Integer.parseInt(values[2]);
            if ((index < 0) || (index > 255))
                throw new IllegalArgumentException("invalid unique index: must be >= 0 and < 256");
            uniqueId = "{";
            for (int i = 0; i < 16; i++, index++)
            {
                if (i != 0)
                    uniqueId += ',';
                uniqueId += index % 256;
            }
            uniqueId += '}';
        }
    };

    // This is for example. In fact these are read from application.properties
    private int localClusterMemberPort = 9992;
    private String clusterMembers = "172.17.0.2:9992:1,172.17.0.3:9992:2,172.17.0.4:9992:3";
}
```

Experiment 3

“Pure” Embedded Tomcat

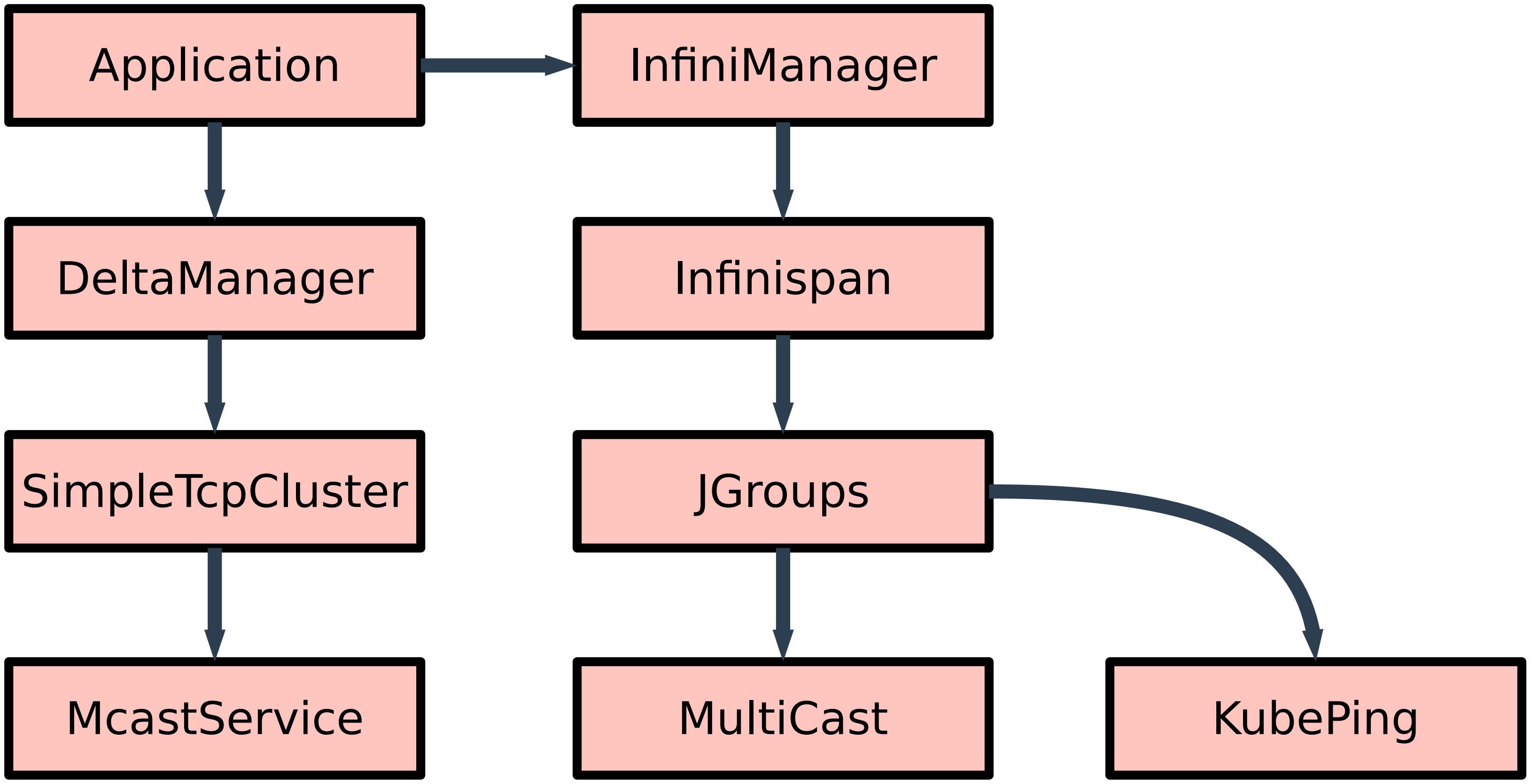
- No *server.xml* file
- Much easier than with Spring Boot

```java
public class Main {

    public static void main(String[] args) throws Exception {
        String contextPath = "" ;
        String appBase = ".";
        Tomcat tomcat = new Tomcat();

        int port = 8080;
        tomcat.setPort(port);
        tomcat.getHost().setAppBase(appBase);
        StandardContext ctx = (StandardContext) tomcat.addWebapp(contextPath, appBase);

        SimpleTcpCluster cluster = new SimpleTcpCluster();
        tomcat.getEngine().setCluster(cluster);
        // Seems like cluster must be added to engine, not context
        //ctx.setCluster(cluster);

        ctx.setName("{CTX}");
        ctx.setDistributable(true);
        ctx.setPrivileged(true);

        cluster.setChannelStartOptions(3);

        DeltaManager manager = new DeltaManager();
        manager.setName("{DELTA}");
        manager.setNotifySessionListenersOnReplication(true);
        cluster.setManagerTemplate(manager);

        GroupChannel channel = (GroupChannel) cluster.getChannel();

        NioReceiver receiver =  new NioReceiver();
        receiver.setPort(9991);
        channel.setChannelReceiver(receiver);

        channel.addInterceptor(new TcpPingInterceptor());
        channel.addInterceptor(new TcpFailureDetector());
        channel.addInterceptor(new MessageDispatchInterceptor());

        StaticMembershipInterceptor membership = new StaticMembershipInterceptor();
        membership.addStaticMember(member(2));
        membership.addStaticMember(member(3));
        membership.addStaticMember(member(4));

        channel.addInterceptor(membership);

        cluster.addValve(new ReplicationValve());
        cluster.addValve(new JvmRouteBinderValve());

        ctx.setManager(manager);

        tomcat.start();
        tomcat.getServer().await();
    }

    private static Member member(int i) {
        StaticMember member = new StaticMember();
        member.setHost("172.17.0." + i);
        member.setPort(9991);
        // Create dummy id = {0, 0, ..., 0, i}
        byte[] id = new byte[16];
        id[15] = (byte) i;
        member.setUniqueId(id);
        return member;
    }

}
```

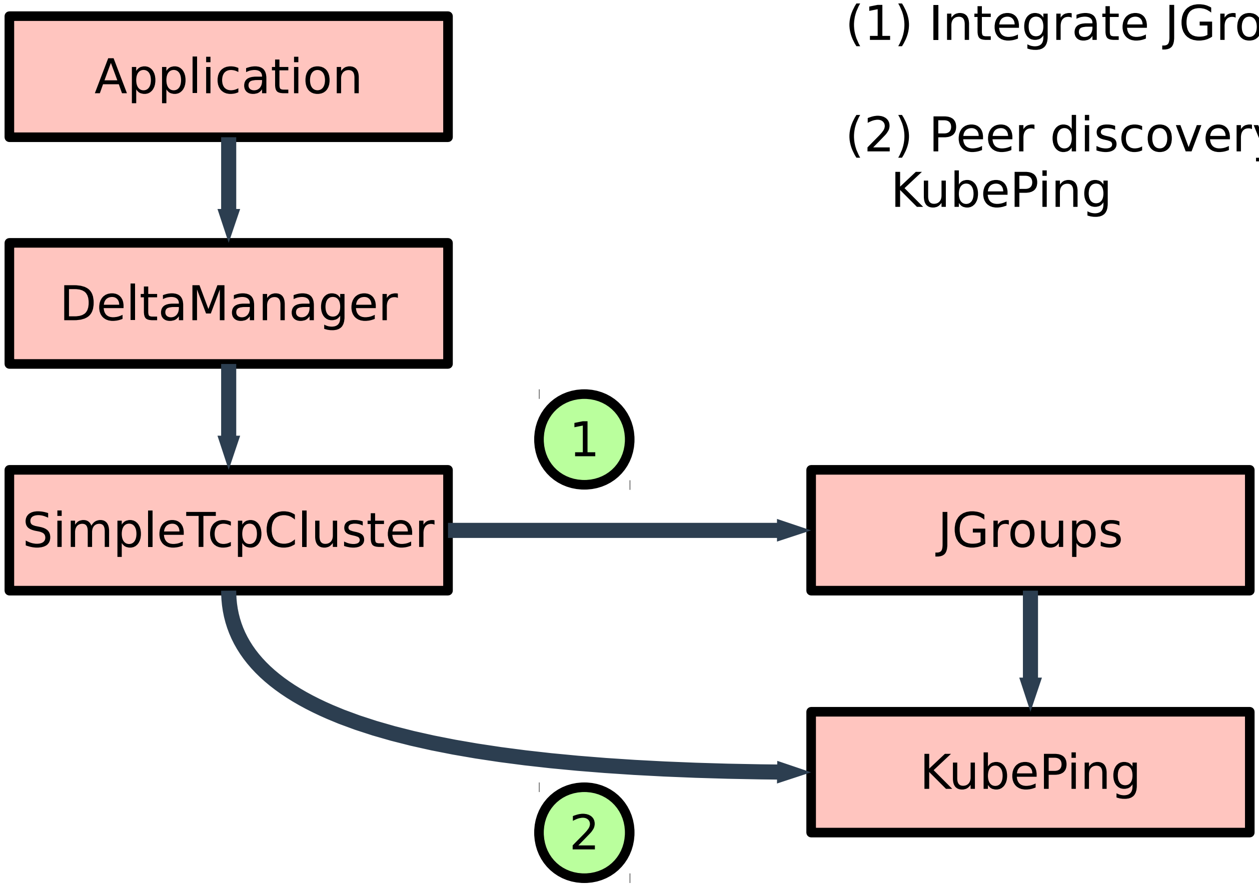
Other Experiments

- **Deploy Embedded Tomcat Application to MiniShift with Fabric8 Maven Plugin**
- **Plan B solution using Spring Session & Redis**
- **Install an OpenShift Cluster** (unsuccessful)
- **Build a test application with Infinispan**
- **...**

Towards a Solution



2 Possible Solutions



(1) Integrate JGroups into Tomcat

(2) Peer discovery directly through KubePing

Planning

Until midterm

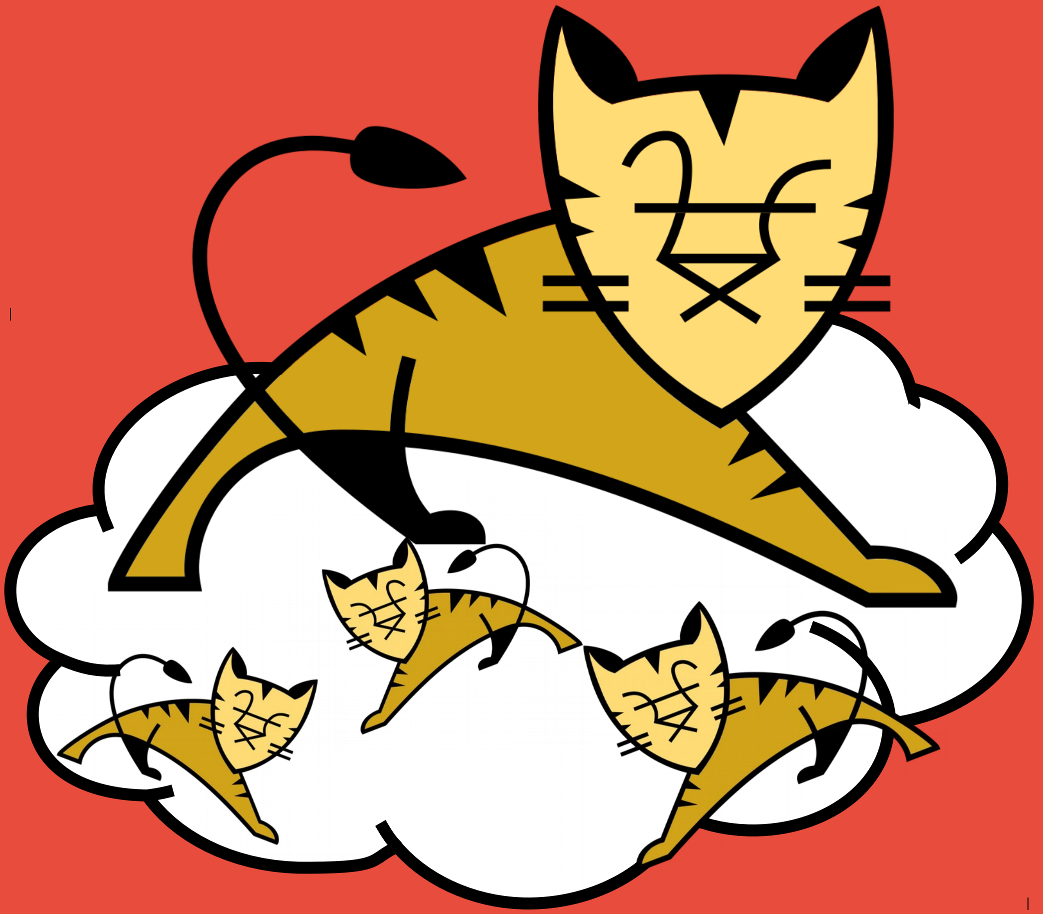
- Read about Tomcat, clustering, Openshift,...
- Write the testing app
- Run the app on a local Tomcat Cluster
- Install and configure Minishift
- Implement Plan A

Midterm to final

If Plan A works:

  - Install Raspberry Pi Cluster
  - Deploy app
- Else:
  - Plan B
- Write documentation & report

Thank you for your attention!



Ismaïl Senhaji, Guillaume Pythoud