

Pivotal®

# Apache Tomcat

## Load-balancing and Clustering

Mark Thomas, 20 November 2014

# Introduction

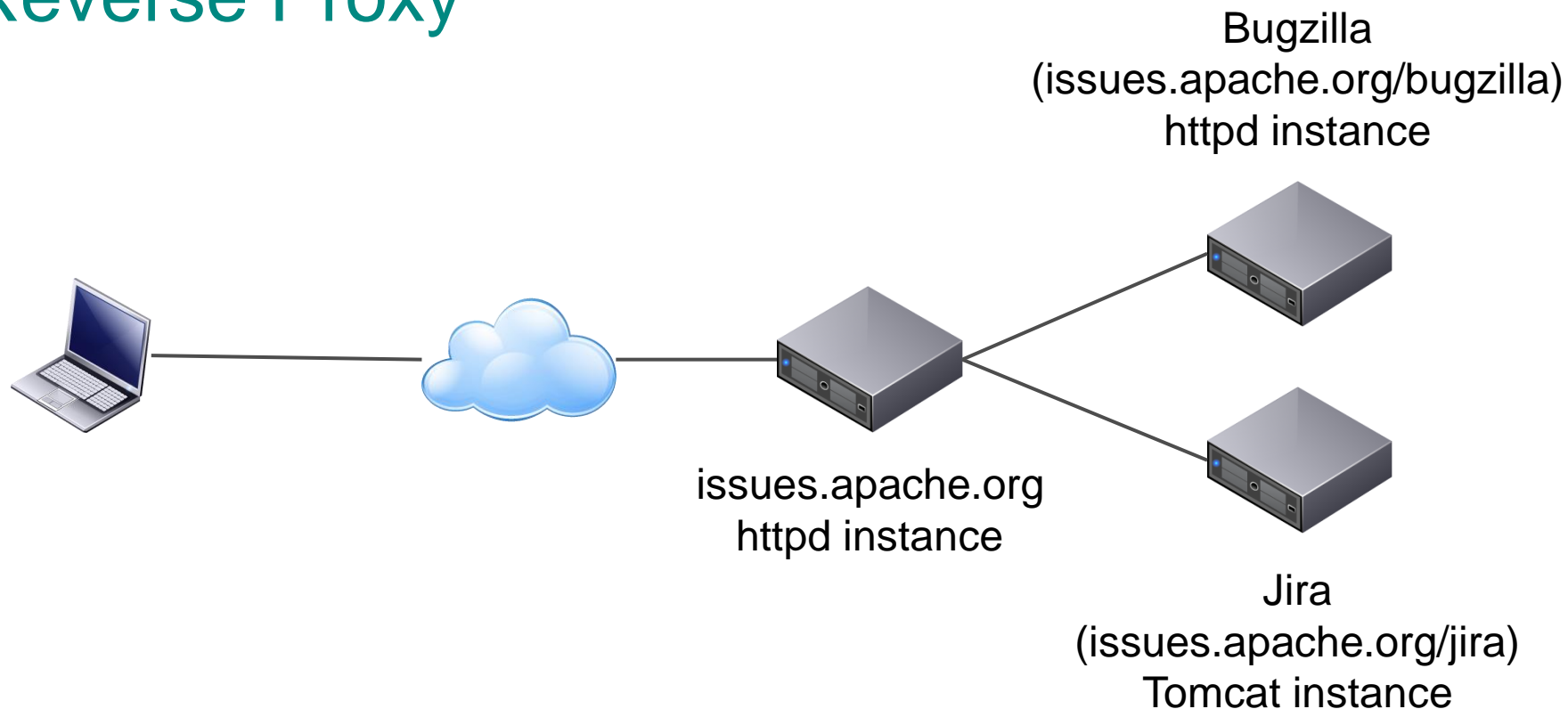
- Apache Tomcat committer since December 2003
  - markt@apache.org
- Tomcat 8 release manager
- Member of the Servlet, WebSocket and EL expert groups
- Consultant Software Engineer @ Pivotal
- Currently focused on Apache Tomcat 9

# Agenda

- Terminology
- Reverse proxies
- Load-balancing
- Clustering

# Terminology

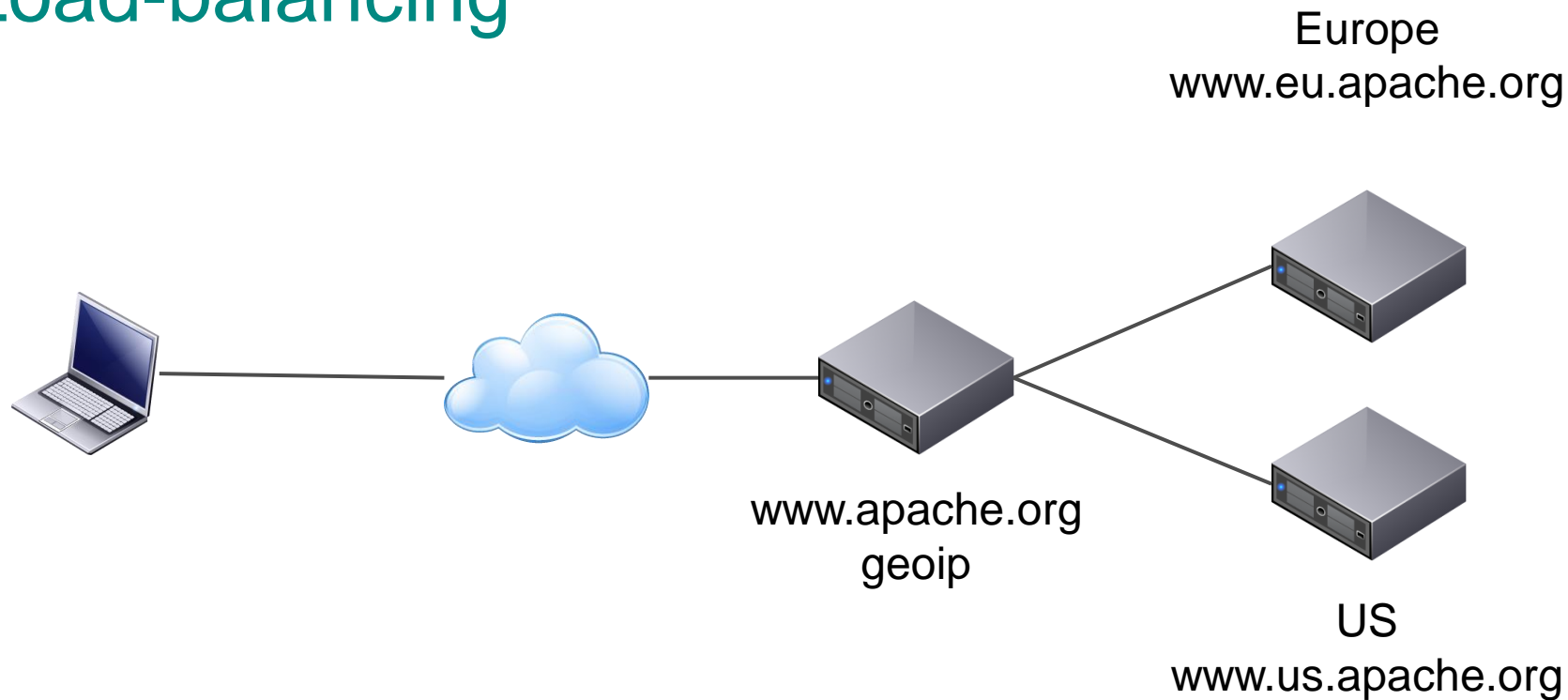
# Reverse Proxy



# Reverse Proxy

- Looks like a single host to the clients
- Usually multiple hosts
- Different services on different hosts
  - May also be geographically distributed
- Can be used to add features
  - e.g. Use httpd as a reverse proxy for Tomcat to add Windows authentication (no longer necessary)

# Load-balancing





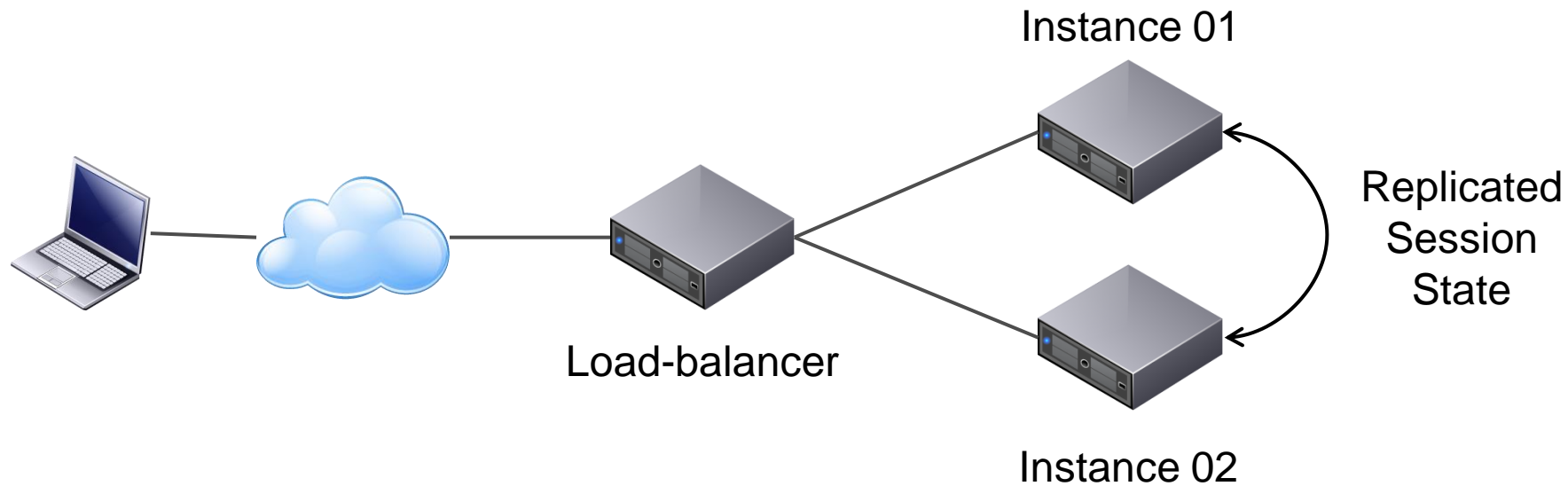
# Load-balancing

- Looks like a single host to the clients
- Multiple hosts
- Each host is the same
- Each host is independent
  - No shared state between the hosts
  - May share common services (e.g. authentication, database)
- Node failure may be visible to users

# Load-balancing

- Lots of options for distributing the load
  - Hardware load-balancer
  - Round-robin DNS
  - Software load-balancer
    - httpd
    - pen
  - geoip

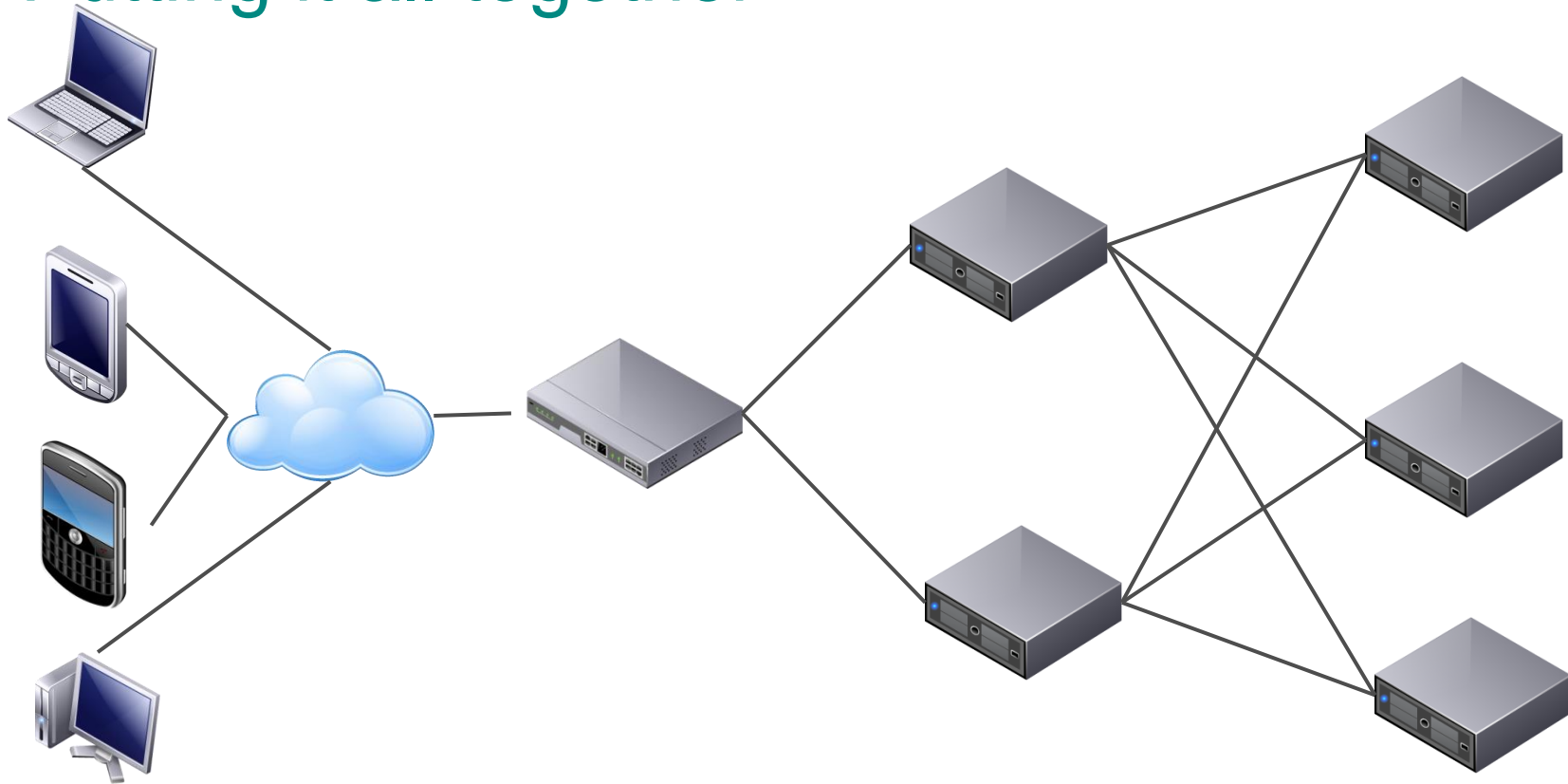
# Clustering



# Clustering

- Load-balancing plus
- Node failure is transparent to users
- This transparency comes at a (usually significant) cost

# Putting it all together



# Reverse Proxies

# Agenda

- Protocol selection
- httpd module selection
- Tomcat connector implementation selection
- Troubleshooting
- Demonstration
  - Time permitting
- Questions

# Protocol Selection

- Two options
  - AJP
  - HTTP
- Best choice depends on circumstances
  - No clear winner
- Both support persistent connections
  - On a fast LAN or the same machine this makes little difference



# Protocol Selection

## AJP

- Not a binary protocol
  - Common headers and values encoded
  - Other values in plain text
  - Request and response bodies in plain text
- Request headers must fit in a single AJP message
  - Default 8192
  - Max 65536

# Protocol Selection

## AJP

- Supports passing of SSL termination information
- Does not directly support encryption
  - IPSec, VPN, SSH tunnel, etc.
- Supports ping to validate connection status

# Protocol Selection

## HTTP

- Clear text protocol
  - Easy to read
- No limit on request header size
- No dedicated ping

# Protocol Selection

## HTTP

- Does not directly support providing SSL termination information
  - Can be added by httpd using custom headers
  - Can be processed by Tomcat using the SSLValve
  
- Supports encryption via HTTPS

# Protocol Selection

## AJP vs. HTTP

- If terminating SSL at httpd and you need SSL the information
  - Use AJP
- If you need to encrypt the httpd to Tomcat channel
  - Use HTTP

# Protocol Selection

## AJP vs. HTTP

- If you need both
  - Use HTTP
  - It is (usually) easier to pass SSL information over HTTP than it is to encrypt AJP
- If you need neither
  - Pick the one you are more familiar with – debugging problems will be easier

# httpd Module Selection

- Avoid
  - mod\_jk2
  - mod\_jserv
  - mod\_webapp
  - anything else not explicitly mention below
- Consider
  - mod\_jk
  - mod\_proxy
  - (mod\_rewrite)

# httpd Module Selection

## mod\_rewrite

- You can replace most of httpd.conf with mod\_rewrite directives
- That doesn't mean that you should
- It is generally more efficient to use the dedicated directive
- There are times (complex load balancing rules) where I've used mod\_rewrite



# httpd Module Selection

mod\_rewrite

- mod\_jk and mod\_proxy can route based on environment variables
- Use mod\_rewrite and/or mod\_setenvif to determine the routing info
- Set the routing configuration with mod\_jk / mod\_proxy

# httpd Module Selection

mod\_jk

- Only supports AJP
- Developed by the Tomcat committers
  - More frequent releases than httpd
  - Features developed in mod\_jk first
- Non-httpd style configuration
- More complex URL mappings are simpler to write
- Binaries only provided for Windows

# httpd Module Selection

mod\_jk

- Doesn't directly support URL re-writing
- Make sure you are using the latest documentation
  - <http://tomcat.apache.org/connectors-doc/>
- The status worker can be used for monitoring and management

# httpd Module Selection

`mod_proxy`

- Supports AJP and HTTP
- Included as standard with httpd
- Uses httpd style configuration
- More complex URL mappings are trickier to write
- Built-in support for URL re-writing (not all use cases)
- Binaries provided for most platforms

# httpd Module Selection

mod\_jk vs/mod\_proxy

- If you need the latest features
  - mod\_jk
- If you have complex mapping rules
  - Consider mod\_jk
- Not on Windows and don't want to have to compile the module
  - mod\_proxy

# httpd Module Selection

mod\_jk vs/mod\_proxy

- If you will be load-balancing
  - mod\_jk's management interface is probably better
- Already using one of these
  - Carry on
  - The costs of changing will probably out-weight the benefits
- If you have a free choice
  - Use mod\_proxy, the configuration style will be more familiar

# Tomcat Connector Selection

- BIO
  - Default for all version to Tomcat 8
  - Removed from Tomcat 9 onwards
  - 100% Java Blocking IO
- NIO
  - Default from Tomcat 8 onwards
  - 100% Java non-blocking IO
    - Waiting for next request
    - Reading HTTP request headers
    - SSL handshake

# Tomcat Connector Selection

- NIO2

- Introduced in Tomcat 8
- 100% Java non-blocking IO
  - Waiting for next request
  - Reading HTTP request headers
  - SSL handshake

- APR/native

- Apache APR based native code with JNI providing non-blocking IO
  - Waiting for next request



# Tomcat Connector Selection

- All connectors block (or simulate blocking) during
  - Request body read
  - Response body write
- SSL
  - BIO, NIO & NIO2 use JSSE
  - APR/native uses OpenSSL
  - OpenSSL is significantly faster
- Sendfile
  - NIO, NIO2 and APR/native support sendfile

# Tomcat Connector Selection

- Comet
  - Will be removed in Tomcat 9
  - NIO, NIO2 and APR/native support Comet
  
- WebSocket
  - All connectors support WebSocket
  - httpd does not support WebSocket when acting as a reverse proxy
  - BIO fakes the non-blocking support

# Tomcat Connector Selection

BIO vs. NIO vs. NIO2 vs. APR/native

- If you use SSL
  - APR/native
- Stability
  - NIO, BIO
- Scalability
  - NIO, NIO2, APR/native

# Troubleshooting

## Thread Exhaustion

- Need to understand threading models
- httpd prefork MPM
  - 1 thread per process
  - MaxRequestWorkers processes
  - Maximum of  $1 * \text{MaxRequestWorkers}$  threads

# Troubleshooting

## Thread Exhaustion

- httpd worker MPM
  - ServerLimit processes
  - ThreadsPerChild threads for each process
  - Maximum of  $\text{ServerLimit} * \text{ThreadsPerChild}$  threads
  
- Thread == concurrent request

# Troubleshooting

## Thread Exhaustion

- Each httpd thread may create a connection to each Tomcat instance
- Therefore, 2 httpd instances each with 400 threads
  - Maximum of 800 connections to each Tomcat instance
  - The connections are NOT distributed between the Tomcat instances
  - Connections are persistent by default

# Troubleshooting

## Thread Exhaustion

- Connections may have low utilization
- BIO requires a thread per connection
- BIO connector may run out of threads even when Tomcat is almost idle

# Troubleshooting

## Thread Exhaustion: Solutions

- Use NIO connector as it is non-blocking between requests
- Don't use persistent connections between httpd and Tomcat
- Ensure each Tomcat instance has  $\geq$  threads than total httpd threads
- Configure timeouts
  - I have seen cases where httpd tried to use a timed out connection
- Use distance to create preferred groups



# Troubleshooting

## Thread Exhaustion: Example

- Reverse proxy for ASF Jira had more threads than Tomcat
- Didn't take much load for Tomcat to run out of threads
- No component was particularly loaded
- Tomcat, Java, network I/O all blamed
- 5 second fix (edit maxThreads in server.xml)
- (OK, and several minutes for Jira to restart)

# Troubleshooting

## Broken Links

- Easiest way to create a lot of hassle for yourself
  - ProxyPass /foo http://localhost:10180/bar
  
- Easiest way to avoid the hassle
  - ProxyPass /foo http://localhost:10180/foo
  
- Don't change the context path

# Troubleshooting

## Broken Links

- Often marketing wants <http://name.com> rather than <http://name.com/app>
- Consider a simple redirect from / to /app
  - /app becomes visible to end users once they use the app
  - Much easier to implement and maintain
- Deploy your application as ROOT
  - Use ROOT##label if you need to add a version number or similar

# Troubleshooting

## Broken Links: What can go wrong

- Redirects
  - Redirect to wrong path
- Cookie paths
  - Cookies are not returned by client
- Links
  - Created for wrong URL
- Custom headers (e.g. Spring MVC)

# Troubleshooting

## Broken Links: Solutions

- Fixing redirects
  - Don't change the context path
  - ProxyPathReverse will fix some but not all HTTP headers
  
- Fixing cookie paths
  - Don't change the context path
  - ProxyPassReverseCookiePath /bar /foo

# Troubleshooting

## Broken Links: Solutions

- Fixing links
  - Don't change the context path
  - `mod_sed`, `mod_substitute`, `mod_proxy_html`
  - Fragile solution and a significant maintenance overhead
- Fixing custom headers
  - Don't change the context path
  - `mod_headers`

# Troubleshooting

## Security Issues

- Need to be careful when terminating HTTPS at httpd
- Tomcat needs to know if request was received over HTTPS
  - Sessions must not transition from HTTPS to HTTP
  - Cookies created over HTTPS must be marked as secure
- mod\_jk and mod\_proxy\_ajp just handle this
- mod\_proxy\_http does not

# Troubleshooting

## Security Issues: Solutions

- Custom headers and the RemoteIpValve
- Two HTTP connectors
  - HTTP traffic proxied to connector with secure="false"
  - HTTPS traffic proxied to connector with secure="true"



# Troubleshooting

## Miscellaneous

- Virtual host selection
  - ProxyPreserveHost on
  
- Client IP based security
  - RemoteIpValve

# Demonstration

Time Permitting

# Questions

# Load-balancing

# Agenda

- Terminology
- Request distribution
- Managing state
- Failover
- Demonstration
  - Time permitting
- Questions

# Terminology

- Sticky sessions
- Without clustering, session is created only on node that handled request
- On next request, the load-balancer could send user to a different node where the session doesn't exist
- Sticky sessions is a mechanism (there are several) that ensures the user returns to the node holding their session

# Request Distribution

- Many ways to select node to handle request
- mod\_proxy
  - Number of requests
  - Number of bytes returned
  - Number of current requests
- mod\_jk
  - As mod\_proxy plus
  - Number of sessions (estimate)

# Request distribution

- Client IP
  - Last octet
- Account number
  - Last digit 0-3, 4-6, 7-9
- Customer type
  - More important customers get priority



# Managing State

- Stateless applications are the simple solution
- Application state
  - State includes authentication
- Options
  - HTTP session
  - Database
  - Request parameters
- Load-balancing is impacted by HTTP state

# Managing State

- Sticky sessions are used for HTTP State
- Session ID
  - Something in the session ID identifies the correct node
  - Users could change this
- Dedicated cookie
  - Users could change this
- Property of client such as IP
  - Beware of ISP that use forward proxies

# Managing State

- Application property
  - Account number
  - Account type
- Often overlaps with load-balancing algorithm

# Failover

- Load-balancer needs to know the state of the nodes
- Nodes need to be taken off-line for maintenance
  - Known in advance
  - Several options
- Nodes will fail
  - Not (usually) predictable
  - Need to be able to detect dynamically
- What is the impact on users?

# Failover

## Maintenance

- More transparent to users means
  - More complex configuration
  - Process takes longer
- Need to drain node of users
  - How long can an HTTP session last?
  - At what point do you stop the node anyway?
- Can Tomcat's parallel deployment feature help?

# Failover

## Unexpected

- Typically there is no separate management channel between Tomcat instances and load-balancer
  - There is with `mod_cluster` from JBoss
- Need to detect failed nodes so failover can happen as early as possible

# Failover

## Unexpected

- Can use a 'failed' request to detect a failed node
- Is a 500 response because the server crashed or because of an application bug?
- Is a timeout because the server crashed or because it is just a long running request?
- Applications that can have long running requests take at least that long to detect failures.

# Failover

## Unexpected

- Monitoring user initiated requests to detect node failure is fragile
- Load-balancer triggered request to known, working, 'simple' page
  - More reliable
  - Still an HTTP request with the associated overhead
- Protocol pings are even faster



# TODO Content

# Demonstration

Time Permitting

# Questions

# Clustering

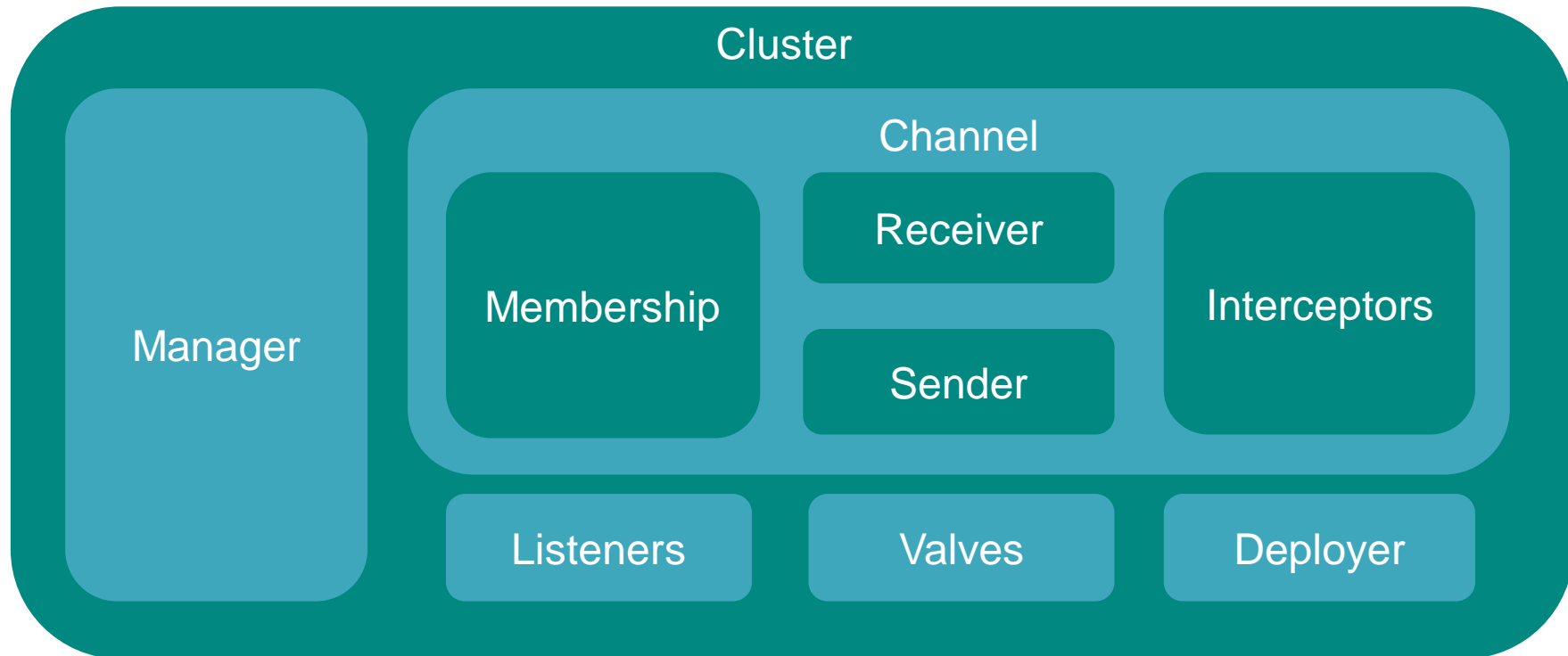
# Agenda

- When to cluster
- Clustering components
- Configuration choices
- Debugging
- Demonstration
  - Time permitting
- Questions

# When To Cluster

- Ideally, never
  - Adds configuration complexity
  - Requires additional processing
  - Debugging is lot harder
- What do you really need?
  - Load-balancing plus sticky sessions
  - If a node fails, sessions will be lost
- Clustering should be the last resort

# Components



# Components

- Cluster
  - Container for all cluster related configuration
  - May be placed within Engine or Host
- Manager
  - Controls how data is replicated between nodes
- Channel
  - Communication between cluster nodes



# Components

- Membership
  - Tracks which nodes joining and leaving the cluster
- Sender
  - Sends cluster messages to other nodes
- Receiver
  - Receives messages from other nodes
- Interceptors
  - Valves for cluster messages

# Components

- Listeners
  - Receive notifications of cluster messages
  - Managers support LifecycleListener
  - Standard session listeners remain available
- Valves
  - Inserted into the request processing pipeline
- Deployer
  - Cluster wide deployment of web applications

# Configuration Choices

## Manager

- Two options
  - Delta manager
  - Backup manager

# Configuration Choices

## Manager

- Delta manager
  - Default
  - Replicates every change to every node
    - Maximum reliability
  - Network traffic proportional to the square of the number of nodes
    - Doesn't scale to large numbers of nodes
  - Fail-over can be to any node

# Configuration Choices

## Manager

- Backup manager
  - Sessions have a primary node and a backup node
    - Need to use sticky sessions
  - Backup node selected on a round-robin basis from all other nodes
  - There is NOT a single backup node
  - Every node knows the primary node and backup node for every session
  - Network traffic proportional to the number of nodes
  - Failover is more complicated

# Configuration Choices

## Manager

### Node A

Primary Sessions:

30\*A

Backup sessions:

10\*B', 10\*C', 10\*D'

### Node B

Primary Sessions:

30\*B

Backup sessions:

10\*A', 10\*C', 10\*D'

### Node C

Primary Sessions:

30\*C

Backup sessions:

10\*A', 10\*B', 10\*D'

### Node D

Primary Sessions:

30\*D

Backup sessions:

10\*A', 10\*B', 10\*C'

# Configuration Choices

## Manager

Node A

Primary Sessions:

30\*A

Backup sessions:

10\*B', 10\*C', 10\*D'

Node B

Primary Sessions:

30\*B

Backup sessions:

10\*A', 10\*C', 10\*D'

Node C

Primary Sessions:

30\*C

Backup sessions:

10\*A', 10\*B', 10\*D'

Node D

Primary Sessions:

30\*D

Backup sessions:

10\*A', 10\*B', 10\*C'



# Configuration Choices

## Manager

- Node D fails
- Sessions will be distributed to other nodes
  - As soon as node failure is detected
- If new node was the backup
  - It becomes the primary
  - A new backup node is selected
  - Session is copied to new backup node



# Configuration Choices

## Manager

- If new node was not the backup
  - It becomes the primary
  - The backup node remains the same
  - Session is copied from the backup node
- Sessions are re-distributed amongst remaining nodes

# Configuration Choices

## Manager

Node A

Primary Sessions:

40\*A

Backup sessions:

20\*B', 20\*C'

Node B

Primary Sessions:

40\*B

Backup sessions:

20\*A', 20\*C'

Node C

Primary Sessions:

40\*C

Backup sessions:

20\*A', 20\*B'

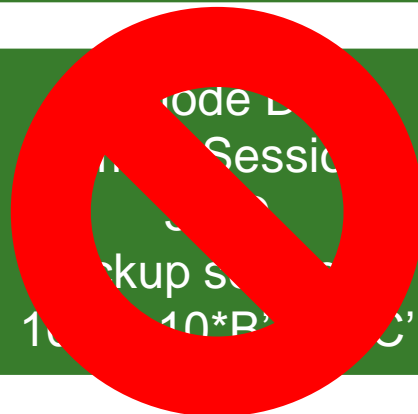
Node D

Primary Sessions:

40\*D

Backup sessions:

20\*A', 20\*B'



# Configuration Choices

## Membership

- Two options
  - Multicast
  - Static

# Configuration Choices

## Membership

- Multicast membership
  - Requires multicast to be enabled on the network
  - Can be difficult to debug problems
  - Scales more easily
- Static
  - Simple to debug
  - Adding nodes gets time consuming as cluster grows

# Configuration Choices

sendOptions

- Delta manager
  - channelSendOptions on Cluster
- Backup manager
  - mapSendOptions on Manager
  
- Synchronous or asynchronous

# Configuration Choices

## sendOptions

- Synchronous

- Request processing does not complete until session data has been sent
- What is meant by sent?
  - On the TCP stack
  - Received by the other node
  - Processed by the other node
- Next request to a different node will see updated sessions

# Configuration Choices

sendOptions

- Asynchronous
  - Request processing continues while session data is sent
  - Next request to a different node may or may not see updated sessions

# Configuration Choices

## Summary

- Manager
  - Delta or Backup
  - (Sticky sessions)
- Membership
  - Multicast or static
- Send options
  - Synchronous or asynchronous



# Debugging

## Cluster Configuration

- Need to know
  - Session ID
  - Current route
  - Which node handled the request
  
- I use a simple JSP page that shows all of the above

# Debugging

## Cluster Configuration

- Quickly test behaviour is as expected
  - Is the route correct for the current node
  - Is load-balancing happening as expected
  - Is fail-over happening as expected
  
- Keep in mind how reverse proxy handles failed nodes

# Debugging

## Application Problems

- Just like trying to debug any other application problem
  - But harder
- Can the issue be replicated in a non-clustered environment?
- Approach depends a lot on the application

# Debugging

## Application Problems

- Network / failover issues
  - Look at the access logs (need session IDs)
  - Look at error logs
  - May need to look at network traffic
- Application issues
  - Logging, logging and more logging
  - Need to be able to fine tune logging

# Demonstration

Time Permitting

# Questions

Pivotal®