Deploying CF in a Cluster and J2EE

About me

- ColdFusion developer for over 10 year
- Adobe Community Expert for ColdFusion
- CTO for Prisma IT in the Netherlands
  - consultancy
  - development
  - hosting
  - training
- Find me at
  - http://jochem.vandieten.net/
  - jochem@prisma-it.com
  - cf-talk@houseoffusion.com

What do I do

- Server stuff
  - CF installation, configuration and troubleshooting
  - Webservers
  - Operating systems
  - Clustering
- Infrastructure
  - Networking
  - Firewalls & loadbalancing
  - DNS
- CF training
- CF development

What do you do?

- developer
- system administrator
- management

Do you have cluster experience?

- No
  - CF Standard
  - CF Enterprise
- Yes
  - CF / J2EE
  - Other

Contents

- Why clustering
- Cluster architectures
- What is J2EE
- Why J2EE clustering
- Why not J2EE clustering
- How to build a J2EE cluster
- How to program for a cluster

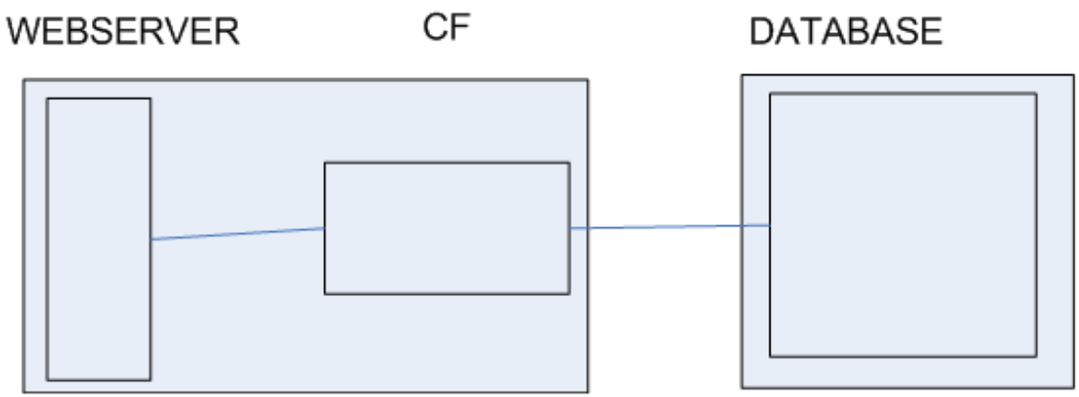
Why clustering

- scalability
- redundancy
- security

Cluster architectures

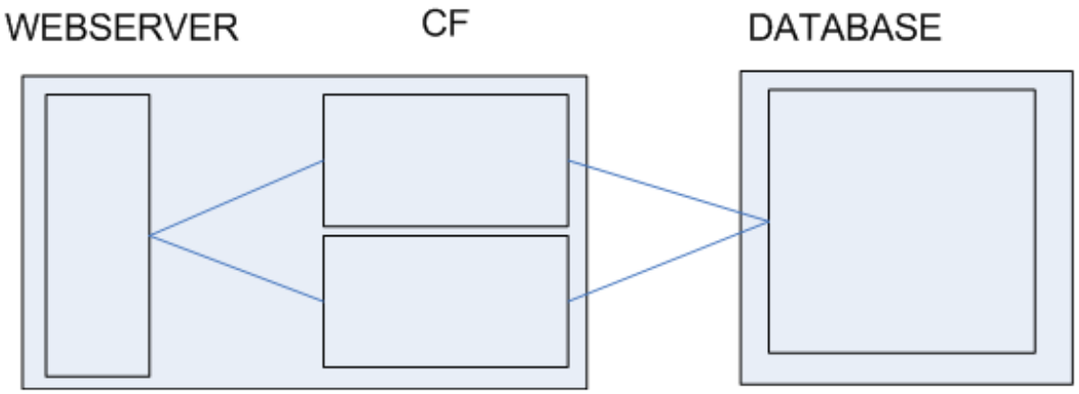
- no cluster
- single server J2EE cluster
- double server cluster
- double server cluster with loadbalancer
- double server J2EE cluster with loadbalancer

Cluster architectures



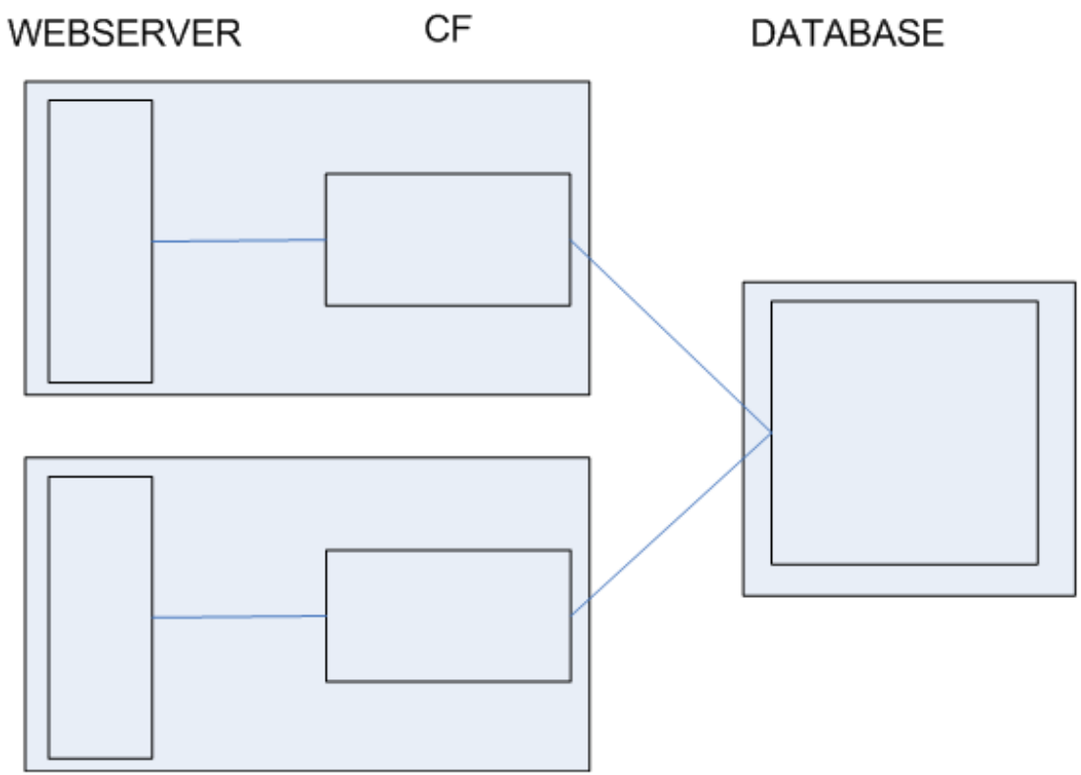
No cluster

Cluster architectures



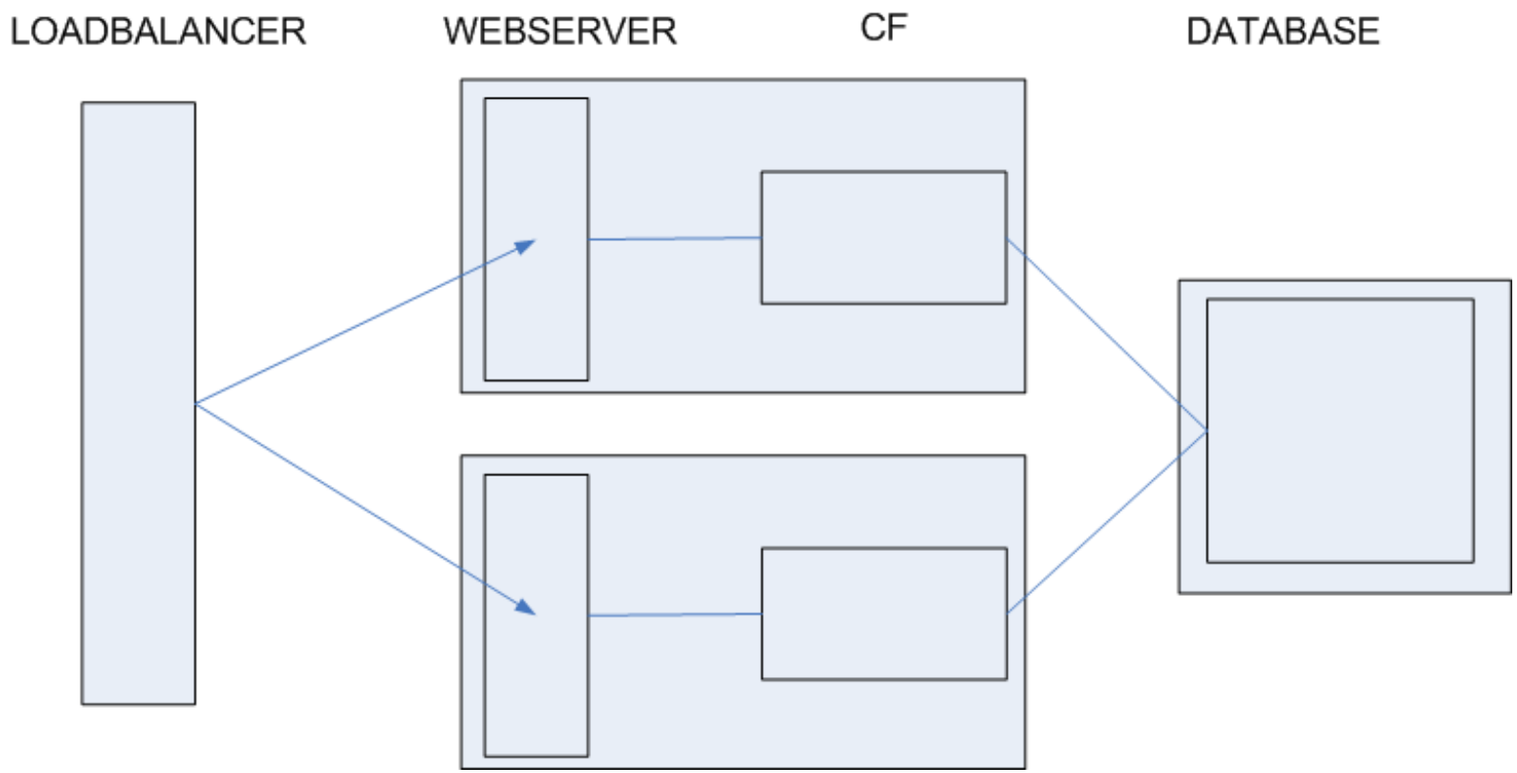
Single server J2EE cluster

Cluster architectures



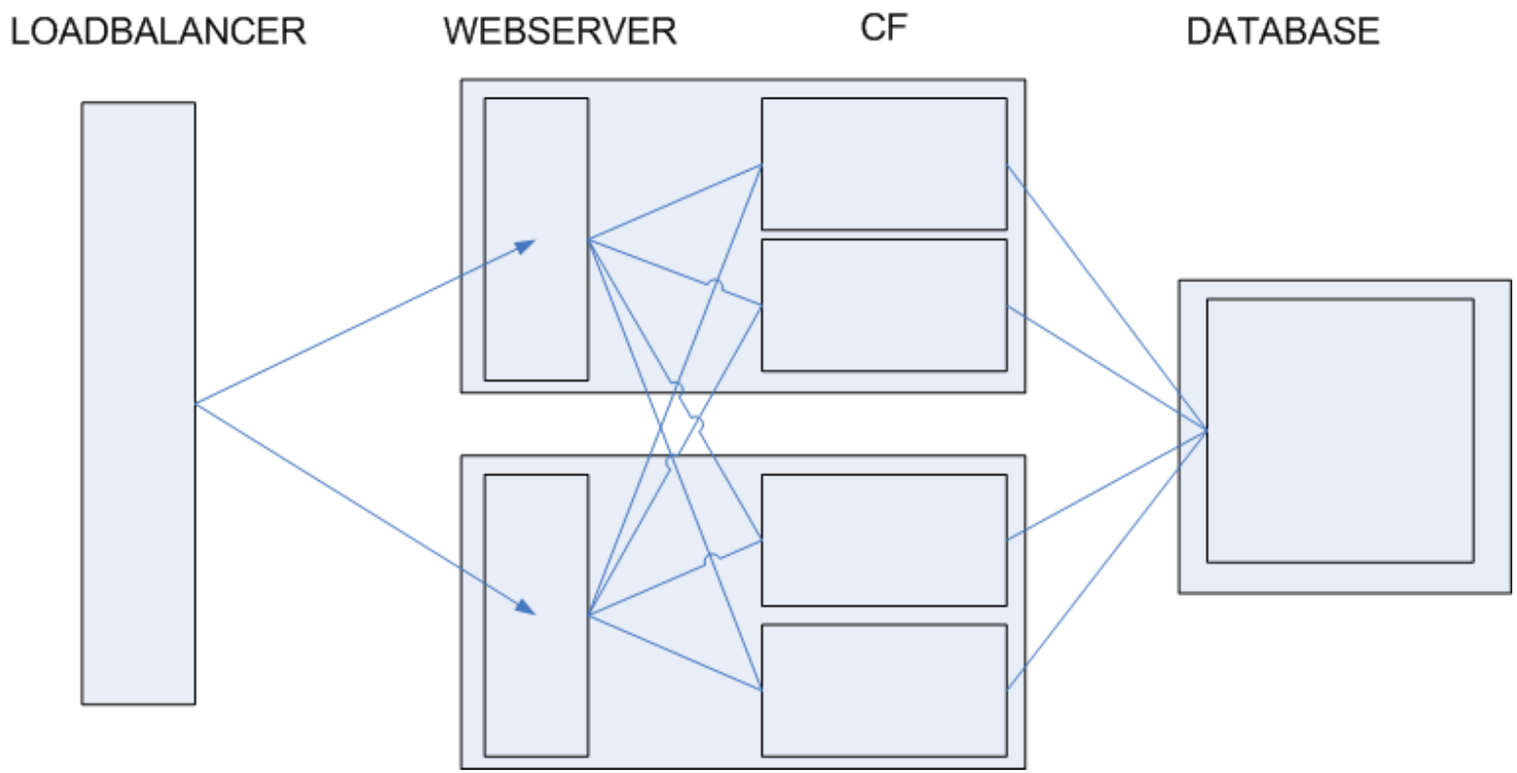
Double server cluster

Cluster architectures



Double server cluster with loadbalancer

Cluster architectures



Double server J2EE cluster with loadbalancer

What is J2EE

JVM
- Java Virtual Machine

JRE
- Java Runtime Environment

JDK
- Java Development Kit

J2EE
- Java 2 Enterprise Edition

JEE
- Java Enterprise Edition

What is J2EE clustering

- loadbalancer is not cluster aware
- webserver is not cluster aware
- clustering in the J2EE server
  - webserver connector is cluster aware

Why J2EE clustering

- it is easy
- it allows you to use session failover
- it is easy
- the least chance of having to change your code
- it is easy

How to build a J2EE cluster

- create your cluster instances
- deploy your code
- switch on clustering

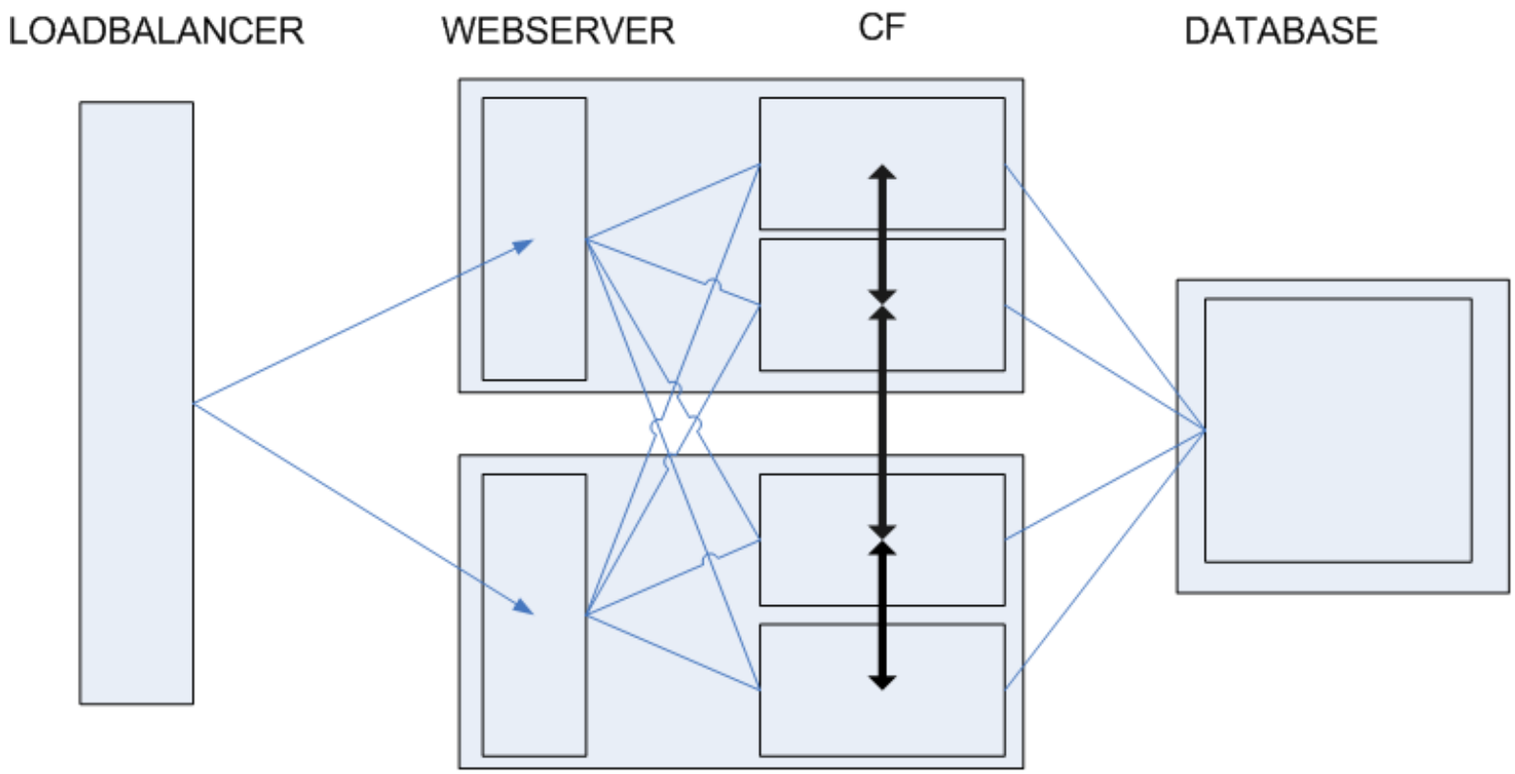
Demonstration

What are sessions

- the web is stateless, for the simplest request all we have is:
  GET /index.cfm HTTP/1.1
  Host: www.prisma-it.com
- we overcome that by adding variables to each request
  GET /index.cfm HTTP/1.1
  Host: www.prisma-it.com
  Cookie: CFID=1100; CFTOKEN=43877514;
- we can keep data in server RAM tied to the session identified by the CFID
- that data is in the session scope
- but since it is in RAM, only one node has access to it
- (we can keep data elsewhere too, like in the database in the case of client variables)

Session failover

- switch on J2EE sessions in CF
- switch on session replication in the J2EE server
- the cluster nodes now replicate session data to eachother



Why not J2EE clustering

- it is usually overkill
- you need CF Enterprise
- you need some precautions in your code (but no more then with other cluster methods)

How to program for a J2EE cluster

- session handling
- assets
- caching
- scheduled tasks
- tags with trouble

Session handling

- switching on session replication is the simple answer
  - failover is automatic
  - performance may be a problem
  - arrays and queries are not replicated
  - don't forget you need J2EE sessions
- working around it is usually possible
  - client variables
  - use sticky sessions

Sticky sessions

- send all requests from one user to the same instance
- requests are identified by the jsessionid
  - you need J2EE sessions
  - /SERVER-INF/connector.properties has the server.id
  - server.id is equal to first 4 bytes of jsessionid
- when the target instance is down, the behaviour degrades to round robin

Assets

- place them in the database
  - single point of failure (if your database is)
  - performance for large files is a problem
- place them on a shared drive
  - single point of failure (if your NAS is)
- place them on each server
  - complicated management
  - diskspace on each server
  - OS features may help

Caching

- many frameworks use an in-memory cache
- there is no convenient mechanism to share caches between nodes
- there is no convenient mechanism to share cache updates between nodes
- evaluate your cache carefully

Scheduled tasks

- CF scheduled tasks depend on HTTP requests
- if you send the HTTP request to the cluster, it can end up at any node
- to make sure the request ends at a specific node
  - use sticky sessions and include a JSESSIONID in the URL
  - fire the request at the builtin webserver

Tags with multiple HTTP requests

Some tags depend on multiple HTTP requests

CFML for a comments form with a CAPTCHA:

```
<form>
<input type="text" name="comment" />
<cfimage action="captcha" text="coldfusion" />
<input type="text" name="code" />
</form>
```

Generated HTML:

```
<form>
<input type="text" name="comment" />
<img src="/CFFileServlet/_cf_captcha/_captcha_img4443716912448663132.png" alt=""
height="36" width="260" />
<input type="text" name="code" />
</form>
```

Tags with multiple HTTP requests

Every tag that depends on multiple HTTP requests needs those requests to arrive at the same node

- cfimage
- cfpresentation
- cfchart
  - specify a shared cache folder as a workaround
- cfdocument
  - use the localUrl attribute
  - use a file://c:/wwwroot/inetpub/...

Demonstration

Code used

```
<html><head>
<title>Cluster Demo Page</title>
</head><body>

<cfoutput>
<h1>#CreateObject("java", "jrunx.kernel.JRun").getServerName()#</h1>
Requests: #session.hitCount++#<br />
</cfoutput>

<cfimage action="captcha" text="ColdFusion" />

</body></html>
```

When not to use J2EE clustering

- when you can avoid it
- 80% of the clusters I built do not need session replication
- sessions are usually used for caching recreateable data instead of true session data
  - on intranets you have integrated security
  - on public sites you have TTP /SAML / HTTP Digest Authentication
- use sticky sessions and accept failover is not gracefull

Checklist

PRISMA·IT

- do you need a cluster
- do you need a J2EE cluster
- cluster your development environment, even if it is your laptop
- enable the builtin webserver
- use J2EE sessions
- use sticky sessions
- include something in your code to identify an instance
- you will need logfile management
- you will need monitoring

J2EE goodies

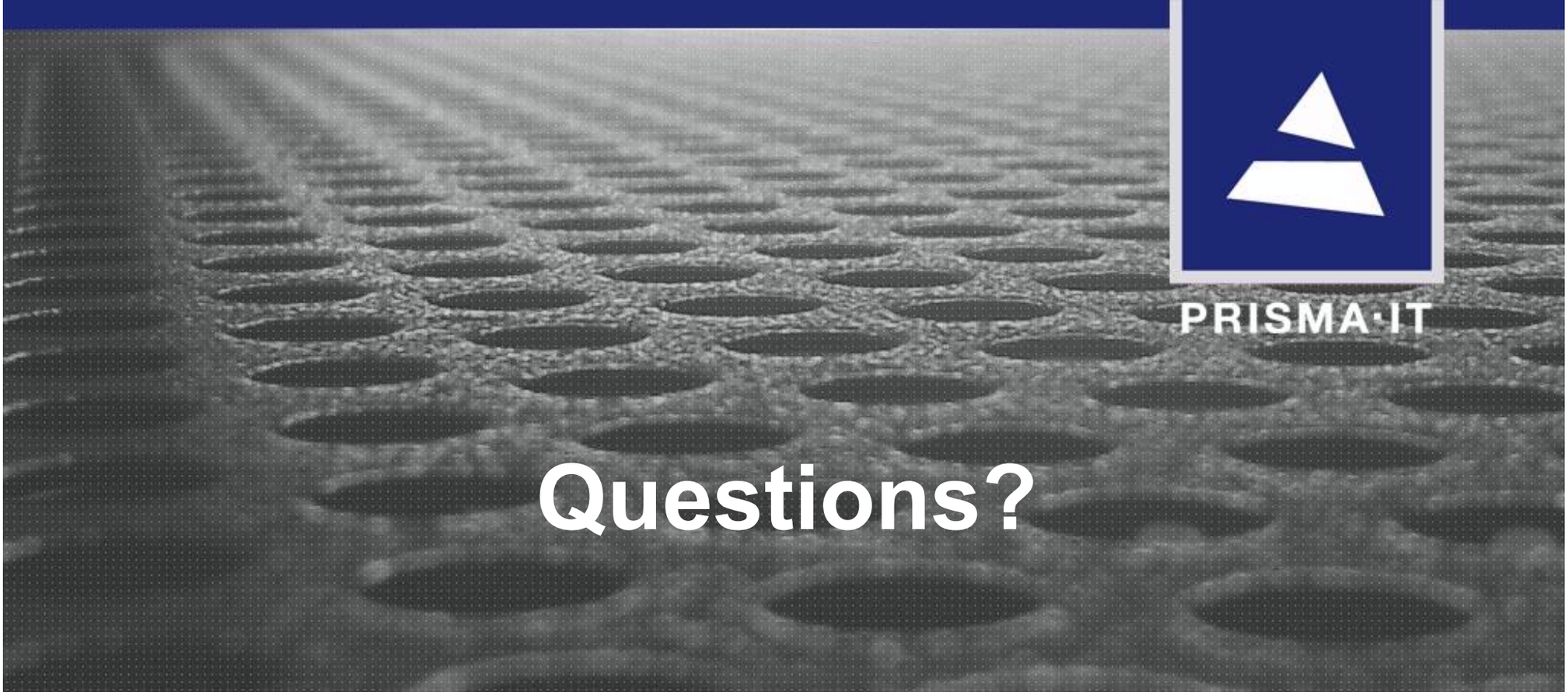
- J2EE is a standard
- you do not have to use the bundled JRun
  - JBoss
  - WebLogic
  - SunOne
  - etc.
- write once, run everywhere
- JMS, RMI etc. are included

Using EAR / WAR files

- J2EE standard archive formats
  - Enterprise ARchive
  - Web ARchive
- not just your code, also your runtime libraries
- code can be compiled
- just a big ZIP file
- easy versioning

EAR file format

- cfusion.ear
  - cfusion.war
    - index.cfm
    - cfc directory
    - image directory
    - js directory
    - WEB-INF
      - web.xml
      - lib
      - cfusion
  - META-INF
    - application.xml



Questions?