APACHECON ASIA 2022

How we use and optimize Tomcat at Alibaba

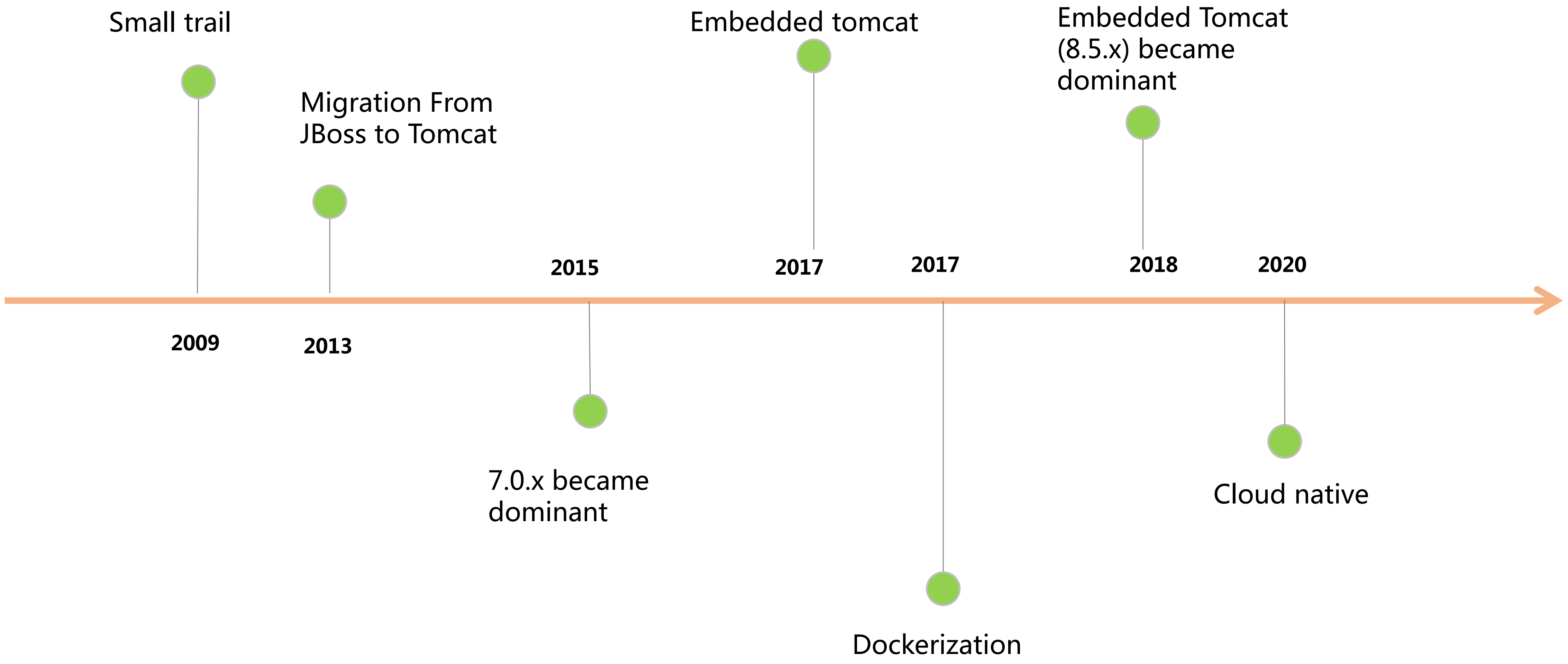
Huxing Zhang

2022.07.29

Biography

- Huxing Zhang (**张乎兴**)
- Staff Engineer @ Alibaba Cloud
- Apache Tomcat Committer since 2016
- Apache Tomcat PMC member since 2017
- Apache Dubbo PMC member since 2019
- Member of ASF since 2019

Tomcat @ Alibaba



How we optimize Apache Tomcat

Dev experience

- Eclipse plugin
- IDEA plugin

Business logic Isolation

- Pandora

Deployment

- Multi-app deployment
- Multi-version deployment
- Multi-tenancy deployment

Diagnostics

- Arthas

Performance Improvement

- App building
- App bootstrap

Observability

- Metrics
- Tomcat monitor
- Tracing
- Async Logging

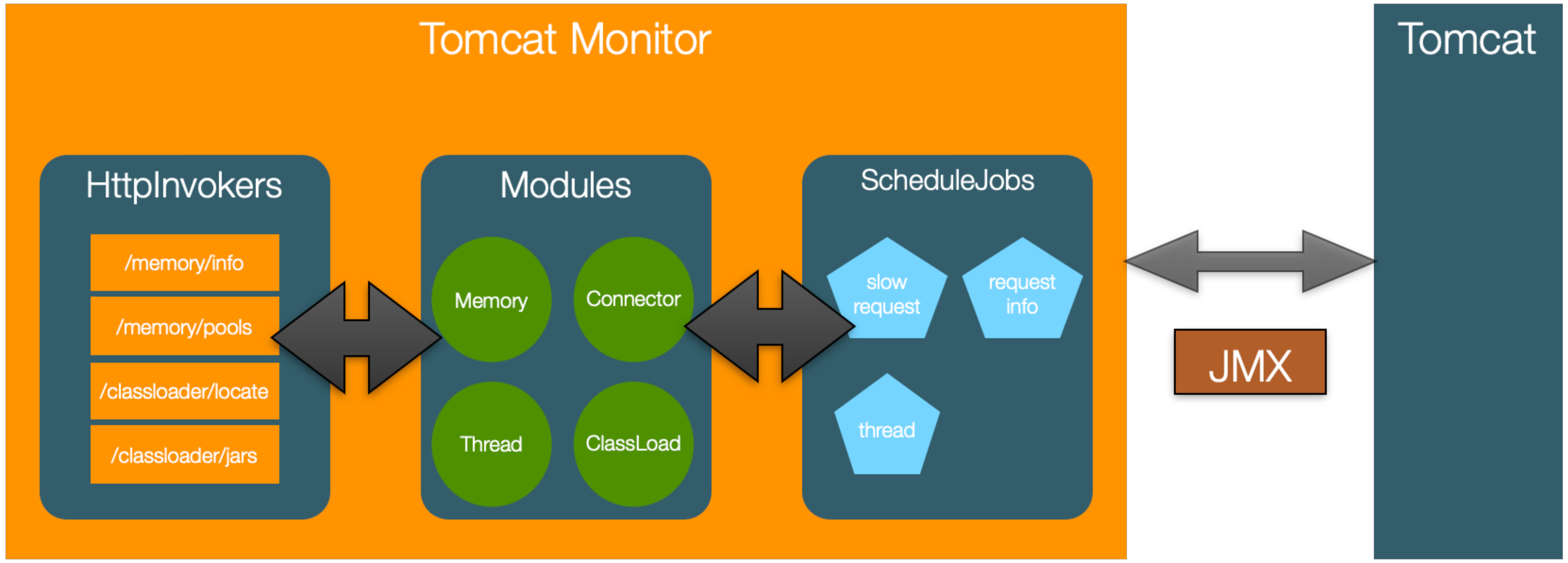
Ops experience

- Auto-migration from Jboss to Tomcat
- App deployment standard

Security

- Security Hardening

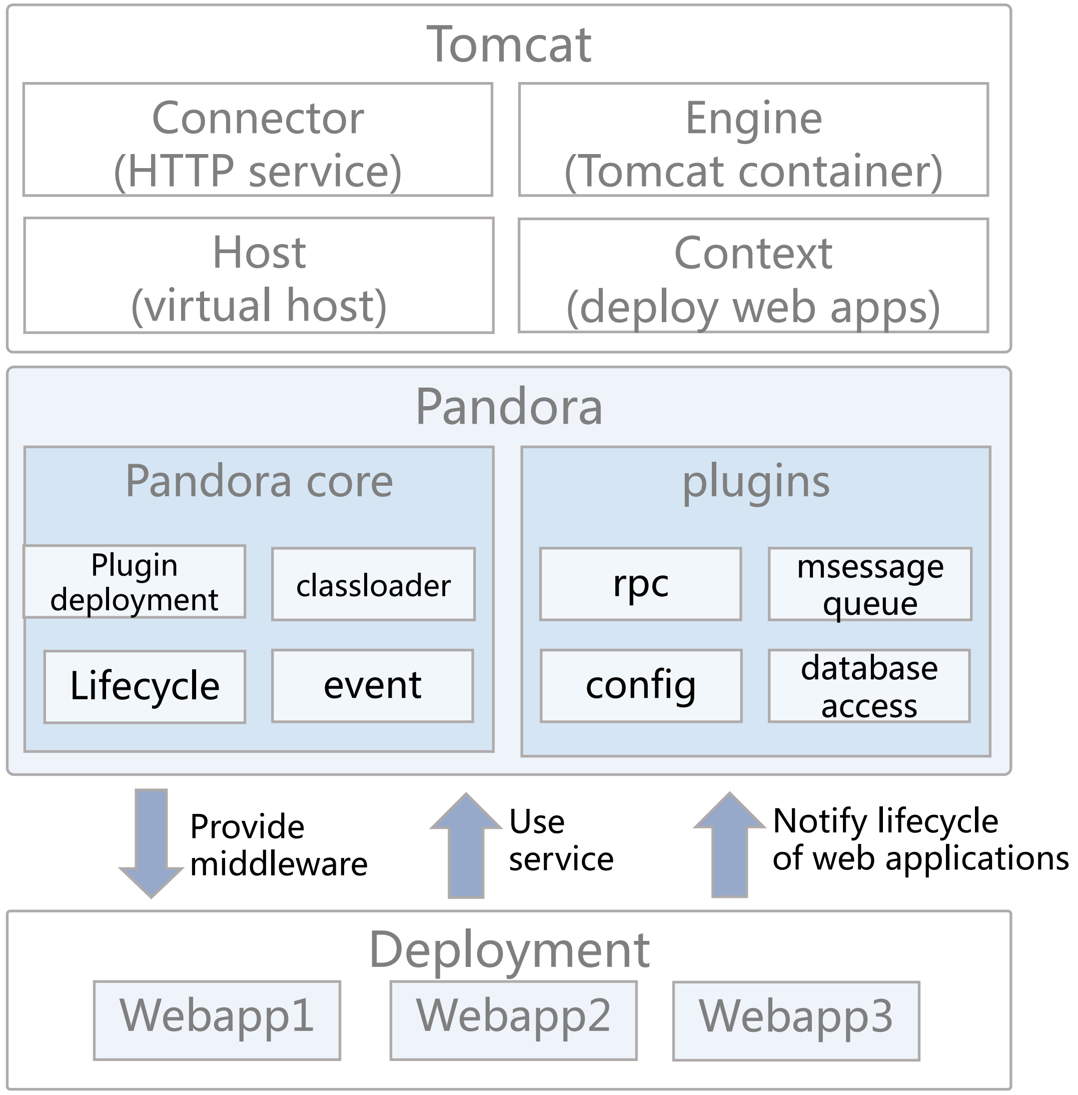
Observability: Tomcat monitor



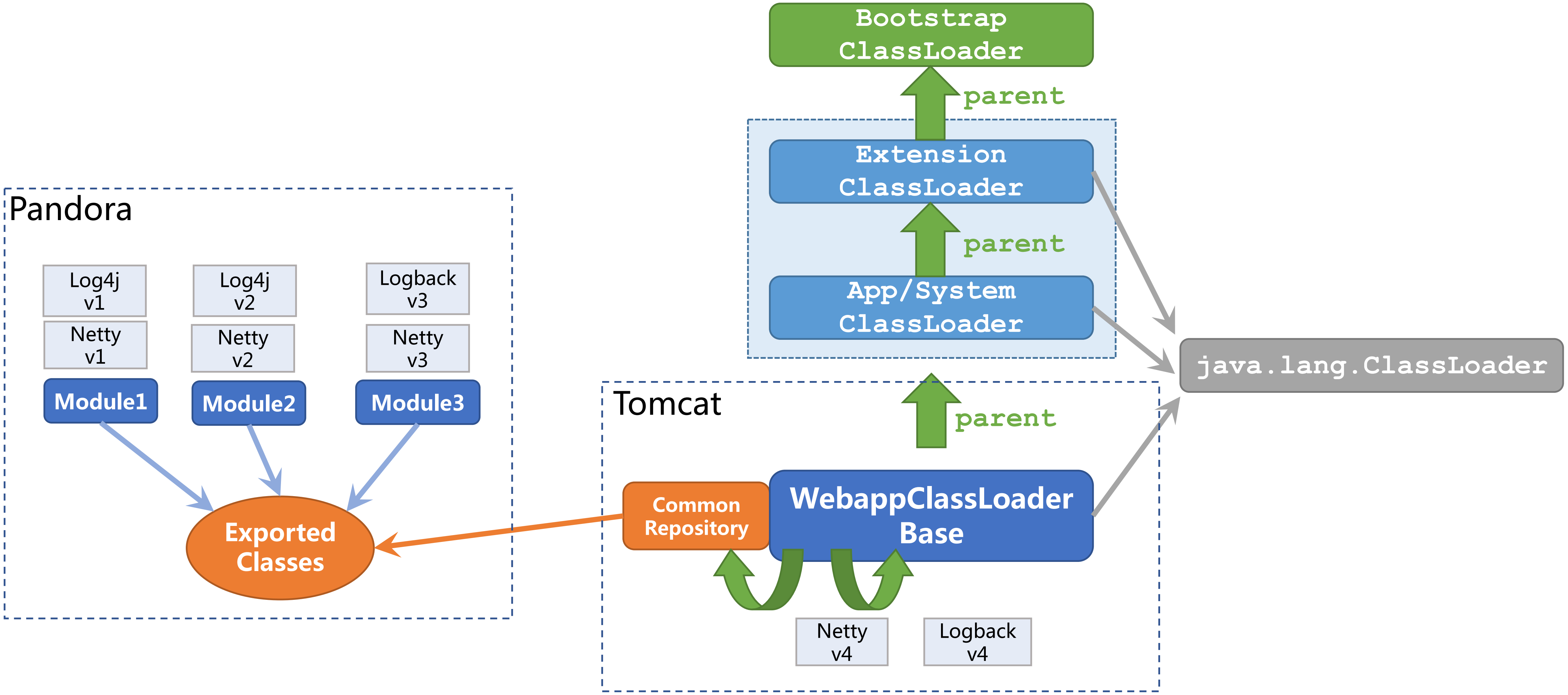
- Metrics for OS, JVM, Tomcat, Middleware, and application
- Classloading: locate which jar file a class is loaded from
- Thread: per thread CPU usage, thread state, etc.
- Connector stats: tomcat thread pool statistics
- Tracing: built-in integration with tracing client using custom Valve

Pandora: light weight isolation container

- Business logic of web applications change fast
- Middleware client has its own dependencies
    - rpc frawework
    - message queue
    - database access
    - cache access
- Necessary to isolate dependencies between web application and middleware clients, and dependencies between middleware clients
- Inspired by OSGi, but lighter weight

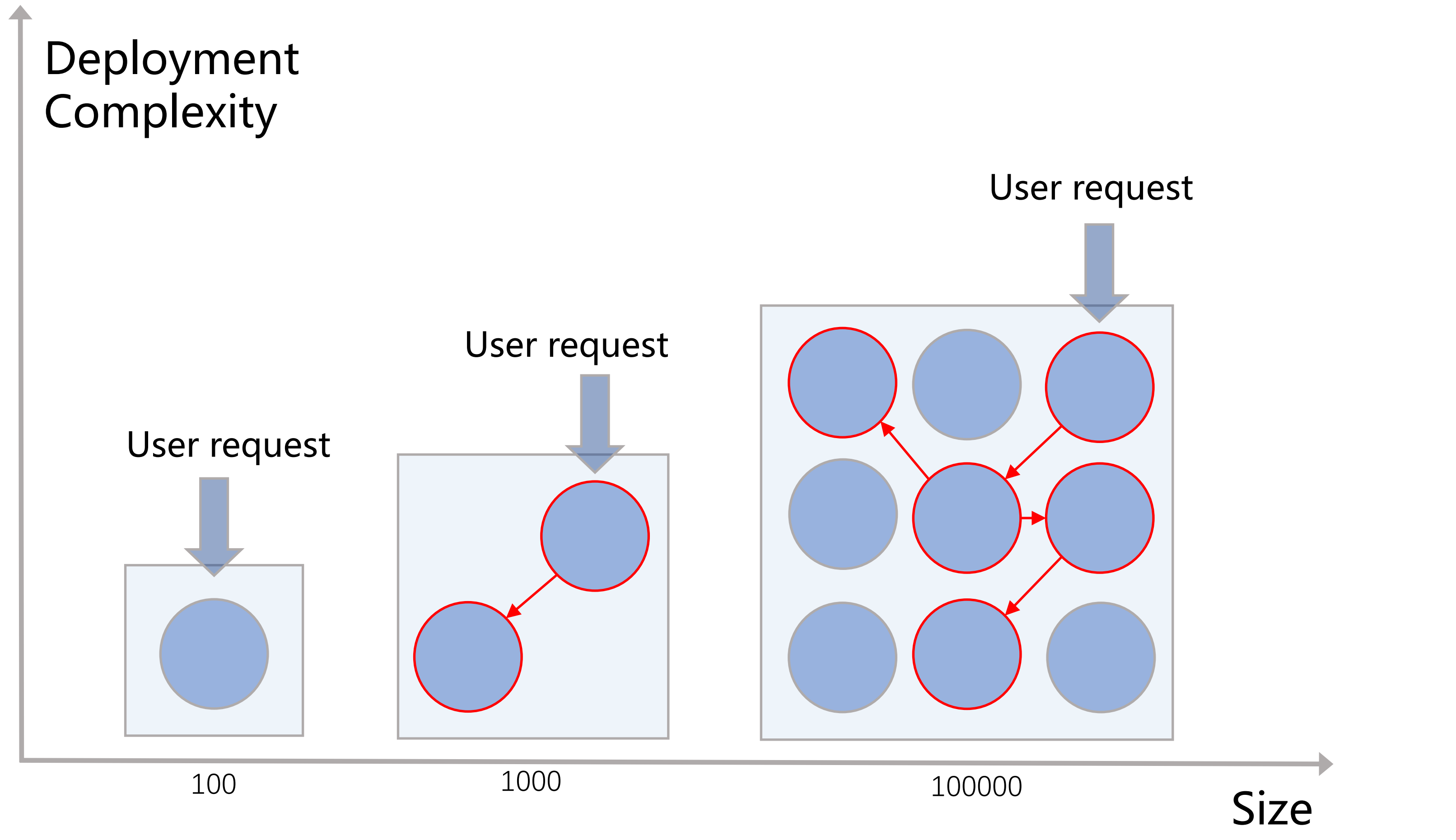


Pandora and Tomcat: class loading

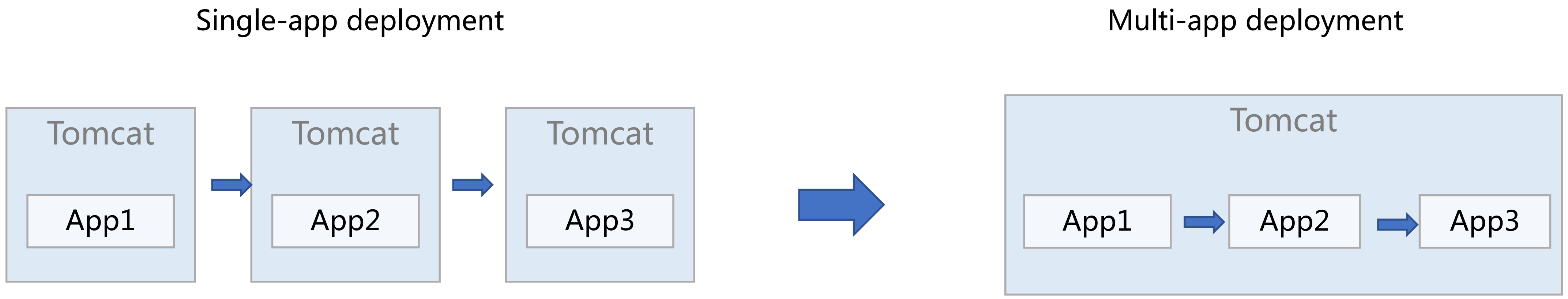


Multi-app Deployment

- Business logic of web applications change fast



Multi-app Deployment

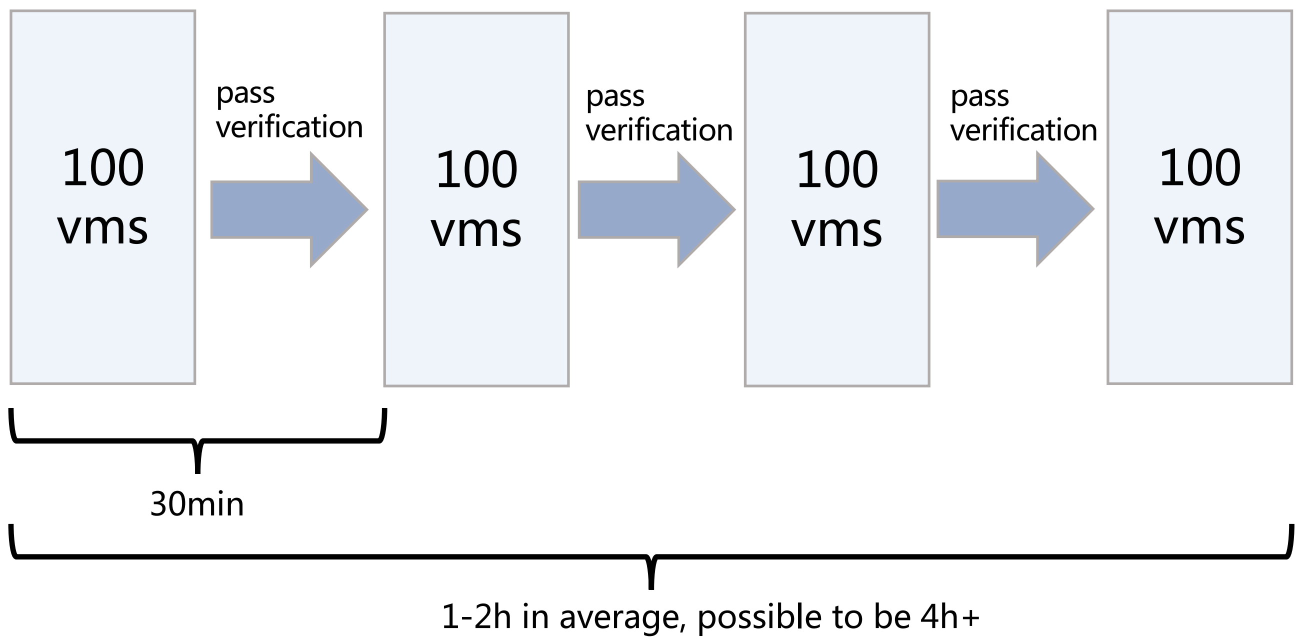


Result

- For core applications: reduce response time
  - 50% qps improvement, rt reduce 50% +
- For long-tail applications: reduce deployment cost

Parallel(Multi-version) Deployment

Assuming deploy an application with 400 vms.



How about increase the number of vms at one time? (e.g. 100 -> 200)

- Reduce overall deployment time
- Fewer available vms, vulnerable to burst traffic

Is it possible to deploy a application without restarting Tomcat?

It just took time to deploy an application, as well as rollback it.

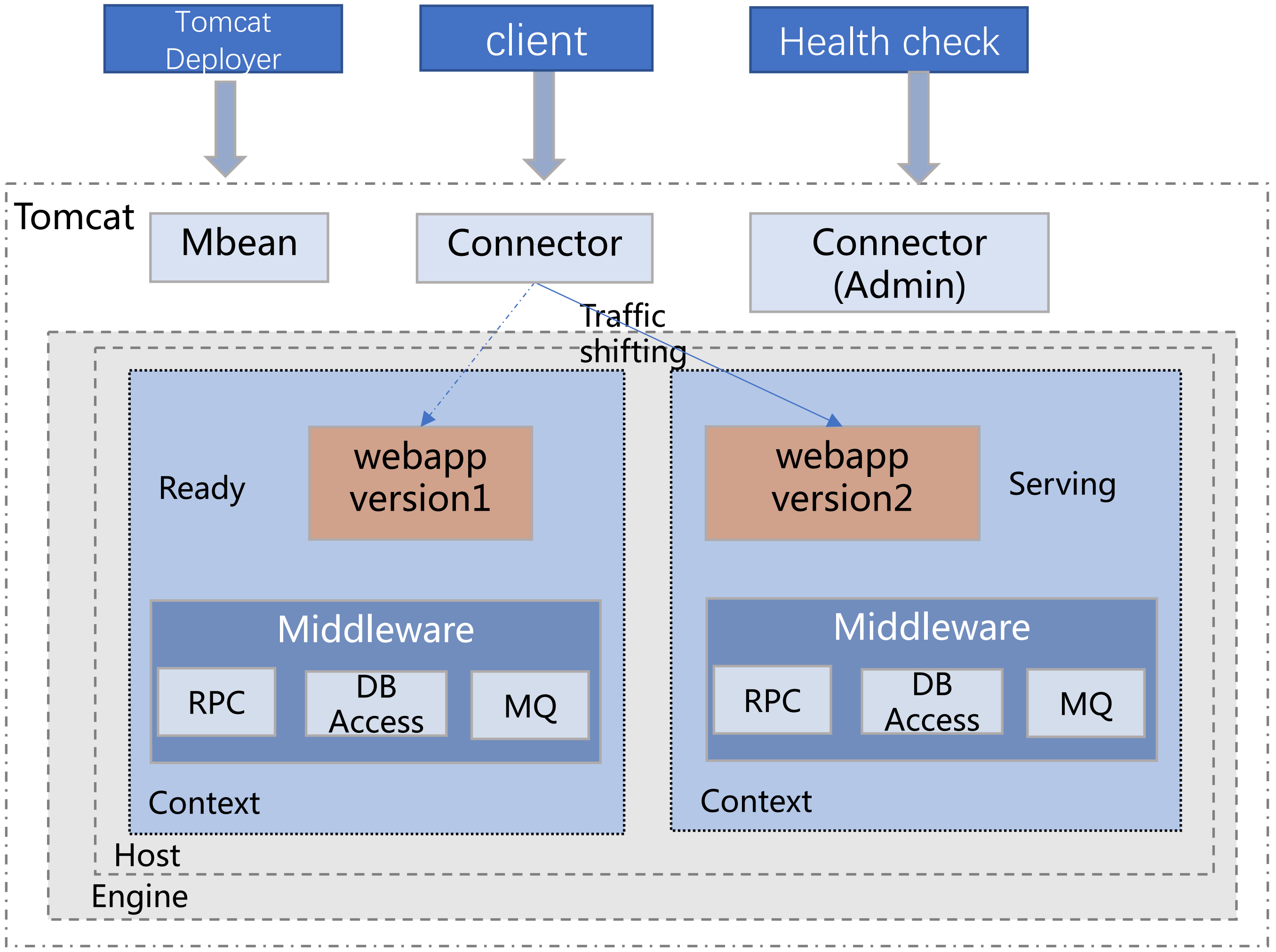
Parallel(Multi-version) Deployment

Enhancement

- New version can be in a ready state unless explicit command to be put to serving state
- Use Tomcat deployer to deploy a new version
- Dedicate port for health check
- Old version can be in a ready state to achieve fast rollback

Challenge

- make sure resources of old version are correctly released.
  - Services registered by rpc framework
  - Listeners of application config
  - MQ consumer
  - DB connections



Parallel Deployment: JVM multi-tenancy

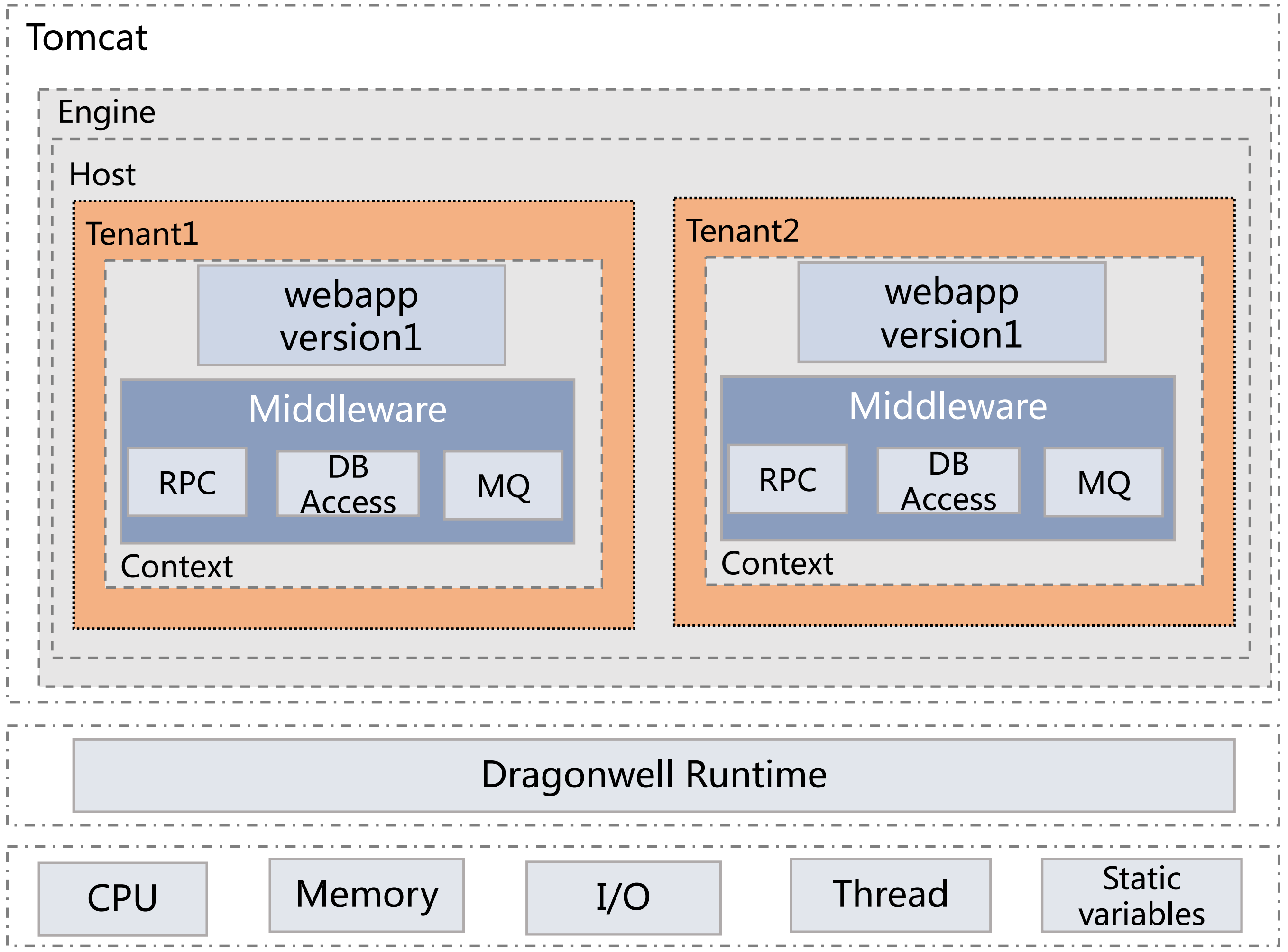
- -XX:+MultiTenant
- -XX:+TenantHeapThrottling
- -XX:+TenantCpuThrottling
- -XX:+TenantDataIsolation

```
TenantContainer tenant =
    TenantContainer.create(new TenantConfiguration()
       .limitMemory(64 * 1024 * 1024)
       .limitCpuShares(1024));


/* code logic runs out of the tenant*/
tenant.run (
      ()-> {
           /* code logic runs inside the tenant */
      }
}

/* code logic runs out of the tenant */

tenant.destroy();
```



- Automatically release the resources when destroying a tenant

Parallel Deployment: Result

Application deployment

1-2h -> 10min

Application rollback

1h -> 1min

Performance

Application Building phase

- Multi-layer docker images for incremental image building
- Parallel maven builds

Application bootstrap phase

- smart annotation scanning
  - Only do if neccessary
- Parallel class loading
- Fast class loading with JarIndex

Fast class loading with Jar index

Bootstrap phase

Read the INDEX.LIST and build the JarIndex Object

```
example.jar

aspectjweaver-1.8.10.jar
org
org/aspectj
org/aspectj/weaver
org/aspectj/weaver/loadtime
…

avalon-framework-4.1.3.jar
org
org/apache
org/apache/avalon
org/apache/avalon/framework
org/apache/avalon/framework/activity
org/apache/avalon/framework/component
…
```

Jar adding phase

For each jar, build a map<String, Integer> where the key is the file name, and the values is the index of the jar file

```
aspectjweaver-1.8.10.jar 0
avalon-framework-4.1.3.jar 1
…
some-other-test.jar 100
```

Class/resource loading phase

contains INDEX.LIST under WEB-INF/lib?

Iterate over each jar

```
aspectjweaver-1.8.10.jar 0
avalon-framework-4.1.3.jar 1
…
some-other-test.jar 100
```

Find the index from the map and load class and resources from the specific jar

```
specific.jar 32
```

Arthas: Trouble-shooting Tomcat on-the-fly

Traditional Way

- Debug production code
  - suspend all the threads
- Reproduce it on test/staging env
  - sometime it is not reproducible
- Add logs
  - time-consuming, test->staging -> production
  - sometime it is not reproducible

The Arthas way

- Trouble shooting production issues on-the-fly.
- No need to modify your code
- No restart of your JVM

Arthas overview

System Info

- dashboard
- thread: find top busy threads
- jvm: view gc details
- View and modify system properties
- View system environment variables

Class / Method / Classloader

- Search specific class
- Search specific method
- Dump class
- Decompile class
- View class loader hierarchy
- Redefine class

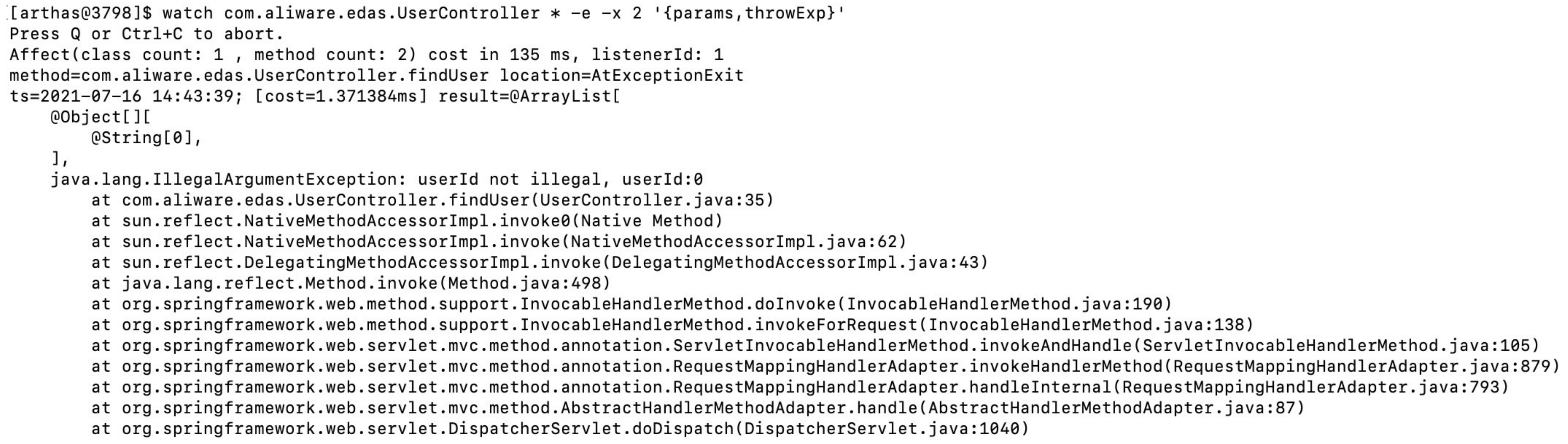
Runtime debugging

- monitor: view invocation stats
- watch: view invocation params and result
- trace: view each invocation cost inside a method
- tt: record invocation context and replay later
- ognl: execute arbitrary command
- performance profiling (flame graph)

https://github.com/alibaba/arthas

Example: view method call params, results and exceptions

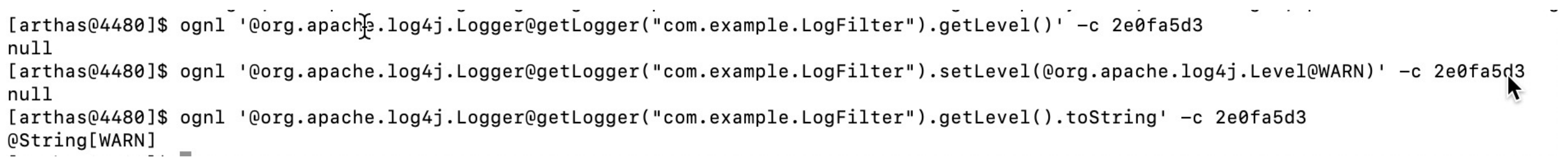
```
$ watch com.example.UserController * -e -x 2 '{params,throwExp}'
```



```
[arthas@3798]$ watch com.aliware.edas.UserController * -e -x 2 '{params,throwExp}'
Press Q or Ctrl+C to abort.
Affect(class count: 1 , method count: 2) cost in 135 ms, listenerId: 1
method=com.aliware.edas.UserController.findUser location=AtExceptionExit
ts=2021-07-16 14:43:39; [cost=1.371384ms] result=@ArrayList[
    @Object[][
        @String[0],
    ],
    java.lang.IllegalArgumentException: userId not illegal, userId:0
        at com.aliware.edas.UserController.findUser(UserController.java:35)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:190)
        at org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:138)
        at org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:105)
        at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:879)
        at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:793)
        at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
        at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1040)
```

Example: adjust log level dynamically

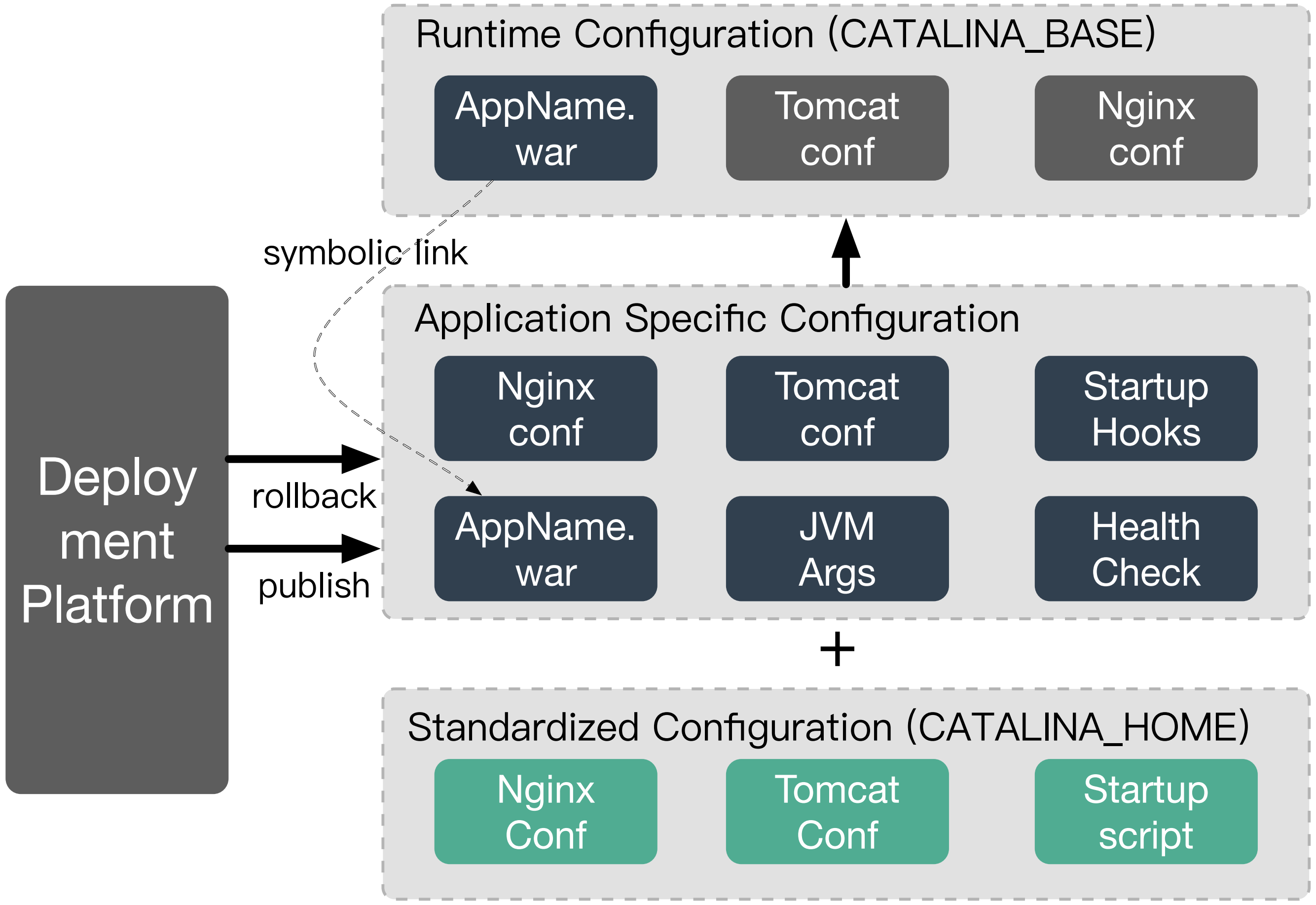
```
$ognl  '@org.apache.log4j.Logger@getLogger("com.example.LogFilter").getLevel()' -
c 2e0fa5d3
$ognl  '@org.apache.log4j.Logger@getLogger("com .example.LogFilter").setLevel(@
org.apache.log4j.Level@WARN)' -c 2e0fa5d3
```



```
[arthas@4480]$ ognl '@org.apache.log4j.Logger@getLogger("com.example.LogFilter").getLevel()' -c 2e0fa5d3
null
[arthas@4480]$ ognl '@org.apache.log4j.Logger@getLogger("com.example.LogFilter").setLevel(@org.apache.log4j.Level@WARN)' -c 2e0fa5d3
null
[arthas@4480]$ ognl '@org.apache.log4j.Logger@getLogger("com.example.LogFilter").getLevel().toString' -c 2e0fa5d3
@String[WARN]
```

Ops: Deployment Standard

One web application per Tomcat instance



Startup script features

- Per startup log rotation for catalina.out
- Auto-fix for incorrect JVM args
- CATALINA_BASE preparation
- Tomcat startup status detection

APACHECON ASIA 2022

Security

- Dynamic cookie management
- Running Tomcat with a non-privileged account
- Change the default directory for deploying web applications, disable ROOT, Docs, Examples, and Manager
- Change SHUTDOWN port and signal
- Disable access logging (available on the reverse proxy side)
- Remove server banner so that Apache-Coyote 1.1 is not returned.

Future work

- GraalVM
- Serverless
- Faster app startup

Thanks

APACHECON ASIA 2022