trivadis
makes IT easier.

Apache incubator ™

Apache Tamaya

Configuring your Containers...

Bio

Anatole Tresch

- Principal Consultant, Trivadis AG (Switzerland)
- Star Spec Lead JSR 354
- Technical Architect, Lead Engineer
- PPMC Member Apache Tamaya

- Twitter/Google+: @atsticks
- anatole@apache.org
- anatole.tresch@trivadis.com

Agenda

- Motivation
- Containers, Confog and Java
- Apache Tamaya
- The API
- Demo & Outlook

Motivation

Some buzzwords... or is there more?

- Microservices
- Containers, Application Servers
- Monoliths
- Java EE
- Evolutionary Architecture
- Configuration

Current Approaches ?

- We hardcode everything
- We configure everything
- We have one configuration system for everything.
- Every solution has its own config.
- Every application has its own config.
- It depends what configuration is...
- We don't know...

But...

- Configuration is an important cross-cutting concern
- ... how ?
- ... when ?
- ... what ?
- ... read-only or can it change ?
- ... decoupled or using some vendor API?
- ... in-process, out-process, local or remote ?
- ...


Apache
incubator ™
Containers, Config and Java
trivadis
makes IT easier.

Java EE Containers

- Well known and established
- Deployment Config only or vendor lock-in
- Config JSR for EE 8 failed ;-(
- CDI for „Application Configuration“ !
- Mostly XML

Docker Containers

- Configuration on deployment by environment properties:

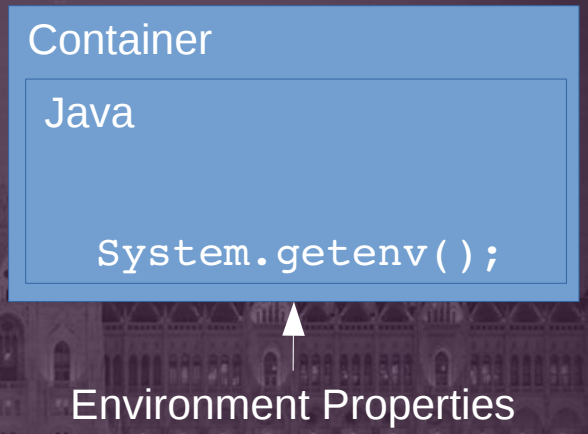
```
docker run -e stage prod  -d -n MyApp user/image
```

- Configuration with Dockerfile:

```
FROM java:8-jre
ADD /hello-drop-1.0.jar //
ADD /hello-config.yml //
EXPOSE 8090 8091
ENV stage prod
ENTRYPOINT ["java", "-jar", "/hello-drop-1.0-.jar",\
    "server", "/hello-config.yml"]
```

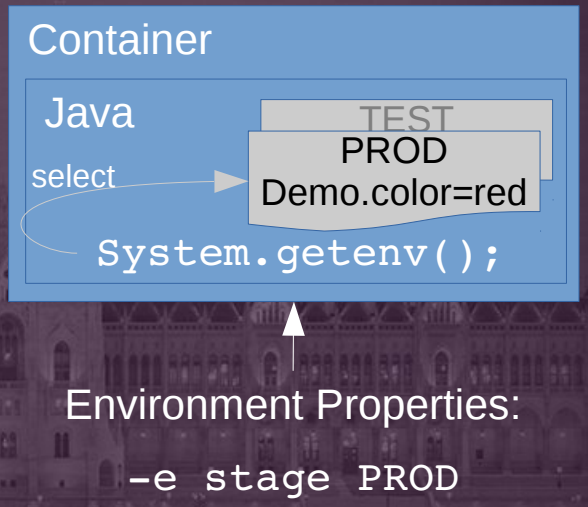
Pattern: Environment Properties

- Configuration by environment properties
  - Well understood
  - Good tool support
  - Static



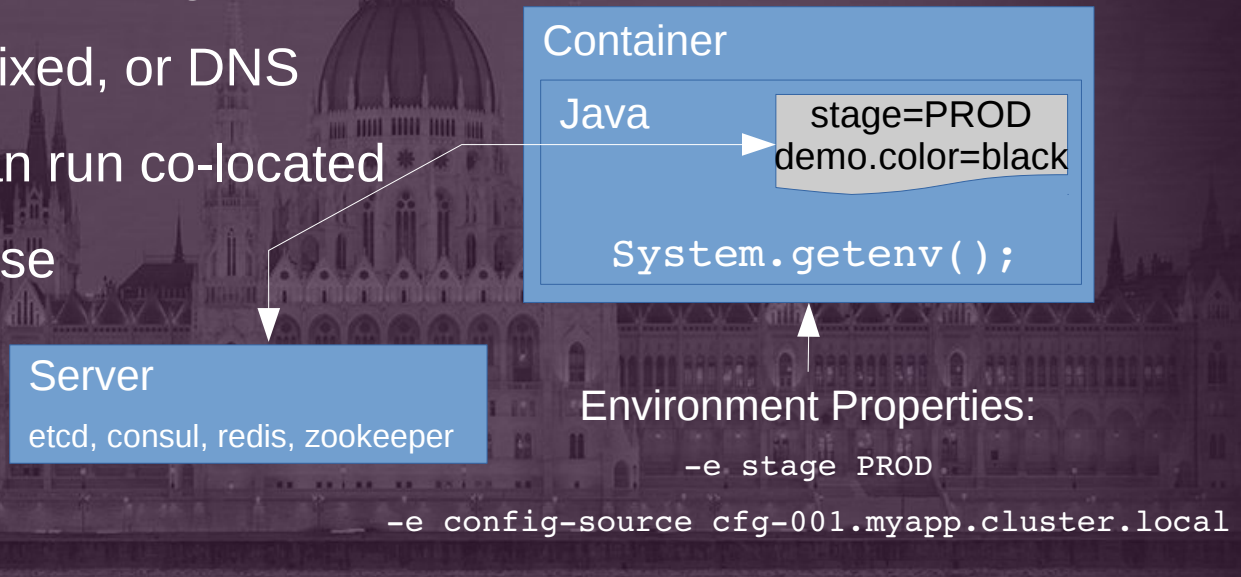
Pattern: Env-Properties + Profiles

- Configuration using profiles
  - Environment Properties used to define the required profile
  - Still static
  - Profiles must be defined during deployment time



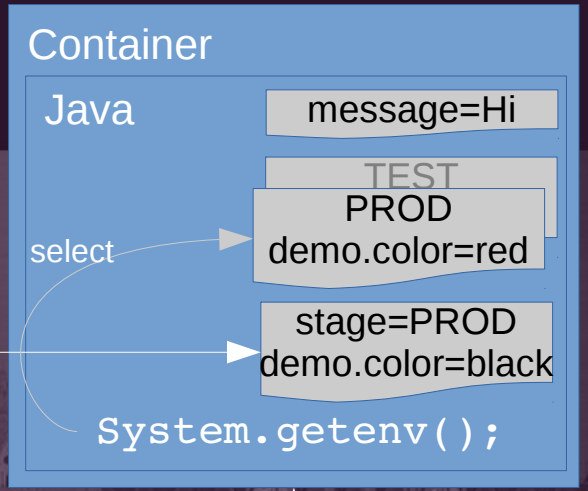
Pattern: Config Server

- Configuration read from Config-Server
  - Environment Properties used to define the required config server
  - Server address fixed, or DNS
  - Config Server can run co-located or somewhere else
  - Vendor APIs



Pattern: Customized

- Configuration combined from multiple sources
  - Defaults on the classpath (profiles)
  - Environment Properties
  - Management Server



Server

etcd, consul, redis, zookeeper

Environment Properties:

```
-e stage PROD
-e config-source cfg-001.myapp.cluster.local
```

Apache Tamaya

http://tamaya.incubator.apache.org

trivadis
makes IT easier.

History of Apache Tamaya

- **2012**: Configuration was voted an important aspect for Java EE 8
- **2013**:
  - Setup of Java EE Configuration JSR failed
  - Standardization on SE Level did not have enough momentum
- **TODAY**
  - Concepts and API are clear
  - Release 0.2-incubating on the way

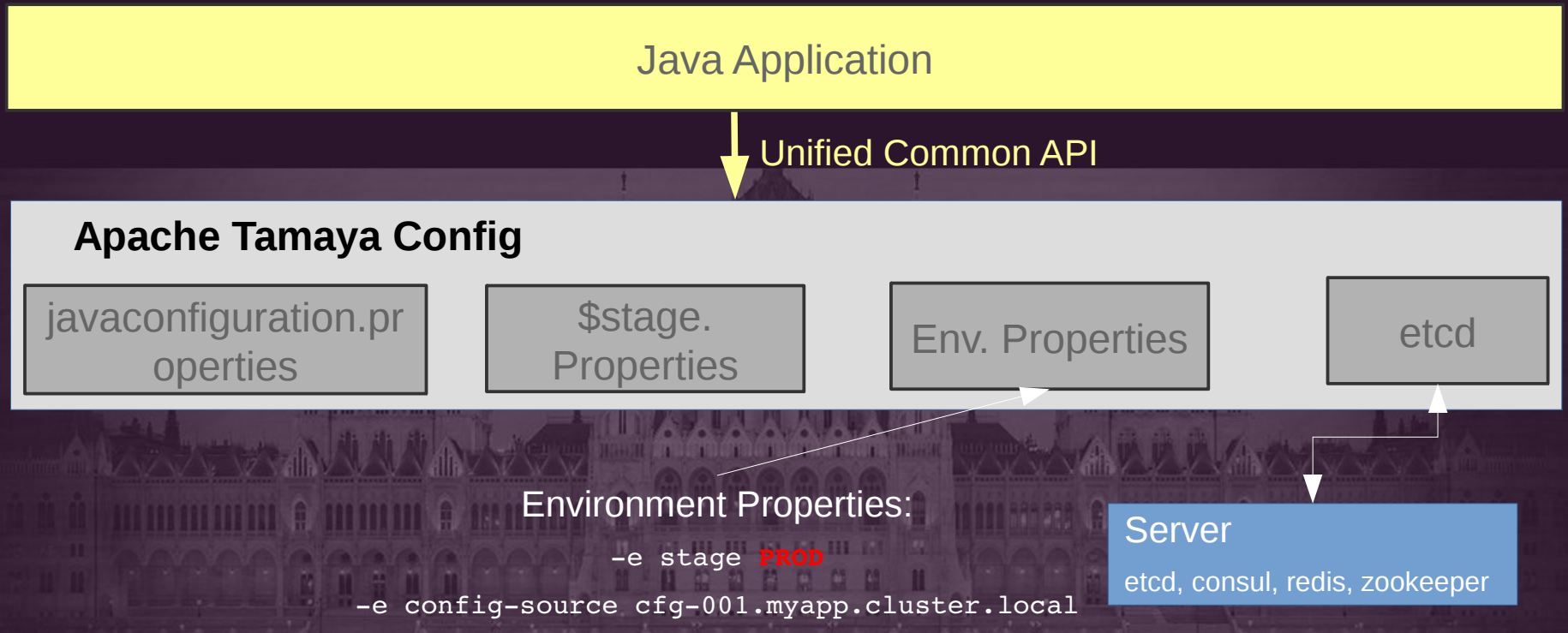
The Objectives of Apache Tamaya

- Define a common API for accessing configuration
  - Minimalistic
  - Flexible, pluggable and extendible
- Compatible with Java 7 and beyond
- Provide a reference implementation
- Provide Extension Modules for additional features
- Build up a community
- Create a Standard!

Decouple your code from...

- **Format**
- **Storage**
- **Lifecycle and versioning**
- **Security**
- **Distribution**
- **Consistency**

Decouple Configuration !



Decouple Configuration !

Java Application

Unified Common API

Apache Tamaya Config

javaconfiguration.properties

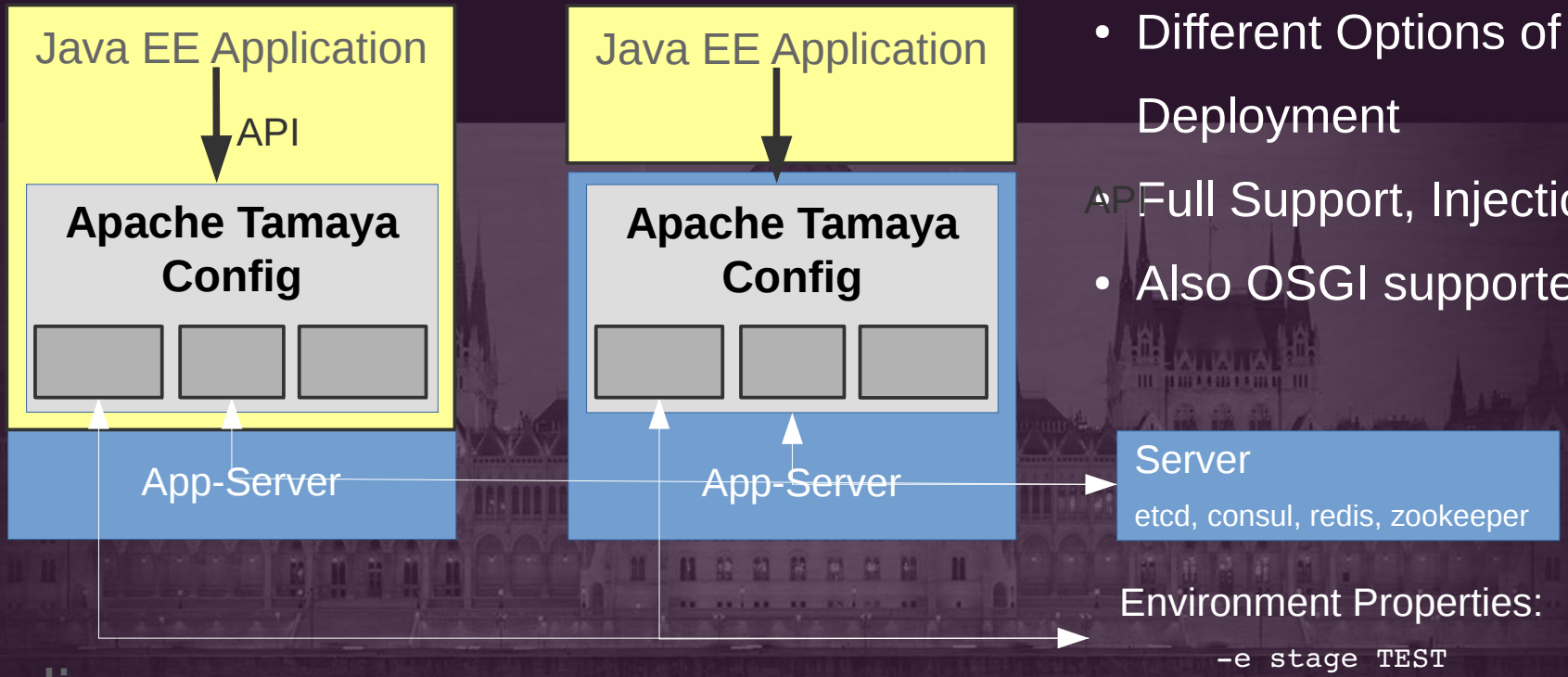
$stage.Properties

Env. Properties

Environment Properties:

```
-e stage TEST
```

How about Java EE ?



- Different Options of Deployment
- Full Support, Injection
- Also OSGI supported

The API

Let's start simple!

- Add dependency

  `org.apache.tamaya:core: 0.2-incubating` (not yet released)
- Add Config to `META-INF/javaconfiguration.properties`
- GO!

```
Configuration config =
                ConfigurationProvider.getConfiguration();

String name = config.getOrDefault("name", "John");
int ChildNum = config.get("childNum", int.class);
```

Configuration Interface


```
public interface Configuration{

  String get(String key);
  String getOrDefault(String key, String defaultValue);

  <T> T get(String key, Class<T> type);
  <T> T get(String key, TypeLiteral<T> type);
  <T> T getOrDefault(String key, Class<T> type, T defaultValue);
  <T> T getOrDefault(String key, TypeLiteral<T> type, T defaultValue);

  Map<String,String> getProperties();

  // Functional Extension Points
  Configuration with(ConfigOperator operator):
  <T> T query(ConfigQuery<T> query);
}
```

Configuration Injection


```
@ConfiguredType(defaultSections=”com.mycomp.tenantAdress”)
public final class MyTenant{

  private String name;

  @ConfiguredProperty(
   defaultValue=”2000”)
  private long customerId;

  @ConfiguredProperty(keys={
   ”privateAddress”,”businessAdress”,
   ”[my.qualified.adress]”
  })
  private String address;
  ...
}
```

```
MyTenant t = new MyTenant();
ConfigurationInjection
    .getConfigurationInjector()
    .configure(t);
```

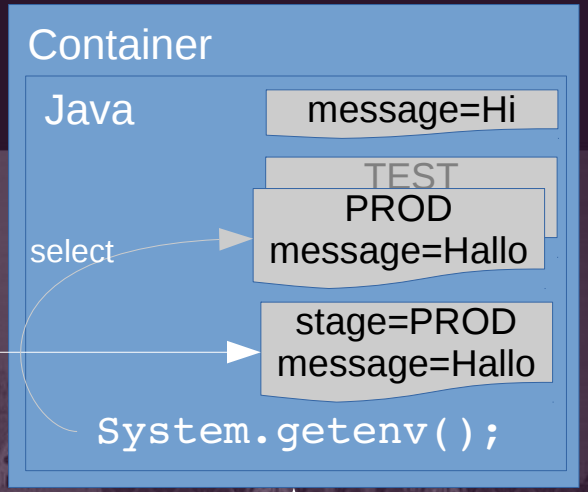
```
@RequestScoped
public class MyClass{
  @Inject
  private MyTenant t;
  …
}
```

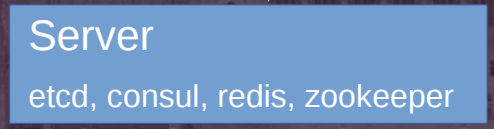
Example: Customized Pattern

- Configuration combined from multiple sources
  - Defaults on the classpath (profiles)

    -> META-INF/javaconfiguration.properties (0)
- Environment Properties
  - -> Already implemented (300).
- Staged Properties (500)
- Etcd Remote Server (1000)

  -> Already implemented, customize defaults



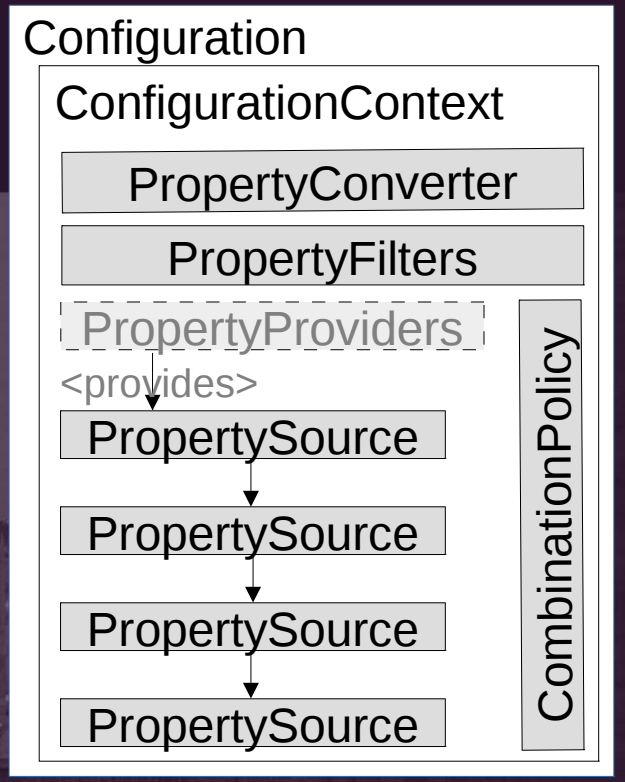


Environment Properties:

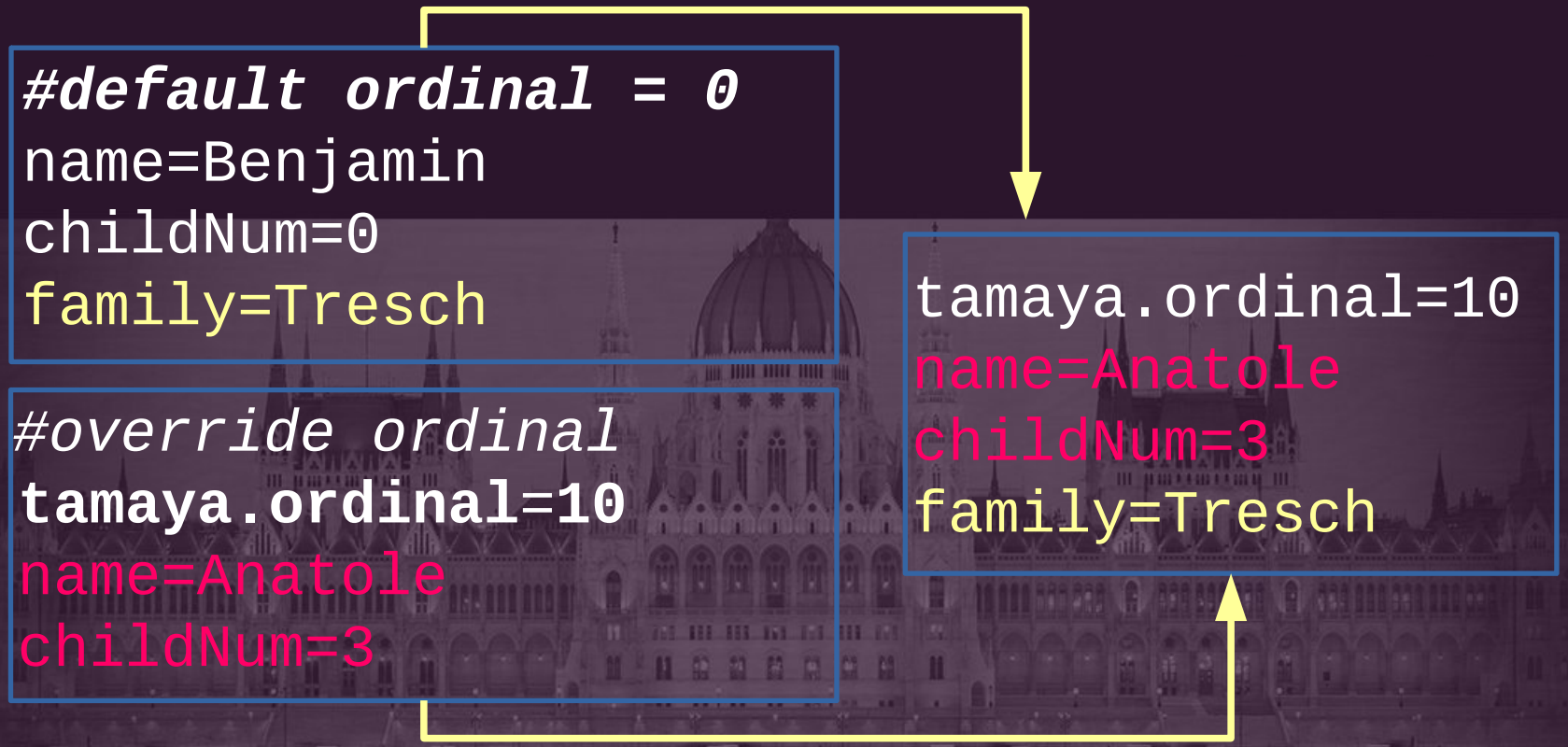
```
-e stage PROD
-e config-source cfg-001.myapp.cluster.local
```

Tamaya Design in 120 Seconds...

1. **`Configuration`** = ordered list of **`PropertySources`**
2. Properties found are **combined** using a **`CombinationPolicy`**
3. Raw properties are **filtered** by **`PropertyFilter`**
4. For typed access **`PropertyConverters`** have to do work
5. **Extensions** add more features (discussed later)
6. **Lifecycle** is controlled by the **`ServiceContextManager`**



Tamaya Design: Overriding



Demo

There is more! - Extension Modules

Extensions: a topic on its own!

- *Tamaya-**spi-support**:* Some handy base classes to implement SPIs
- *Tamaya-**functions**:* Functional extension points (e.g. remapping, scoping)
- *Tamaya-**events**: Detect and publish ConfigChangeEvents*
- *Tamaya-**optional**: Minimal access layer with optional Tamaya support*
- *Tamaya-**filter**: Thread local filtering*
- *Tamaya-**inject-api**: Tamaya Configuration Injection Annotations*
- *Tamaya-**inject**:* Configuration Injection and Templates SE Implementation (lean, no CDI)
- *Tamaya-**resolver**:* Expression resolution, placeholders, dynamic values
- *Tamaya-**resources**:* Ant styled resource resolution
- *Format Extensions:* yaml*, json, ini, … including formats-SPI
- Integrations with **CDI, Spring, OSGI, Camel, etcd**
- *Tamaya-**classloader-support**:* Managing Tamaya Services within Classloading Hierarchies
- *Tamaya-**mutable-config**: Writable ConfigChangeRequests*
- *Tamaya-**server**:* REST/JSON Configuration Server
- *Tamaya-**remote***: Integrate Tamaya Server resources as PropertySource
- *Tamaya-**model***: Configuration Model and Auto Documentation
- *Tamaya-**collections***: Collection Support
- ...

* work in progress

So what is Tamaya about ?

- A Complete thread- and type-safe Configuration API
- Compatible with all major runtimes
- Simple, but extendible design
- A myriad of extensions
- Small footprint

Why you would need Tamaya?

trivadis
makes IT easier.

- Stop Reinventing the wheel
- Stay Focused on what to configure, not how!
- Reduce Redundancies and Inconsistencies
- Document your configuration
- Integrate with infrastructure instead of rewriting config code
- Decouple from configuration backends

And now ?

„It is your turn !“

Links

- Project Page: http://tamaya.incubator.apache.org
- Twitter: @tamayaconfig
- Blog: http://javaeeconfig.blogspot.com
- Presentation by Mike Keith on JavaOne 2013: https://oracleus.activeevents.com/2013/connect/sessionDetail.ww?SESSION_ID=7755
- Apache Deltaspike: http://deltaspike.apache.org
- Java Config Builder: https://github.com/TNG/config-builder
- Apache Commons Configuration: http://commons.apache.org/proper/commons-configuration/
- Jfig: http://jfig.sourceforge.net/
- Carbon Configuration: http://carbon.sourceforge.net/modules/core/docs/config/Usage.html
- Comparison on Carbon and Others: http://www.mail-archive.com/commons-dev@jakarta.apache.org/msg37597.html
- Spring Framework: http://projects.spring.io/spring-framework/
- Owner: http://owner.aeonbits.org/

Q&A

trivadis
makes IT easier.

Thank you!

Anatole Tresch
Trivadis AG
Principal Consultant
Twitter/Google+: @atsticks
anatole@apache.org
anatole.tresch@trivadis.com