

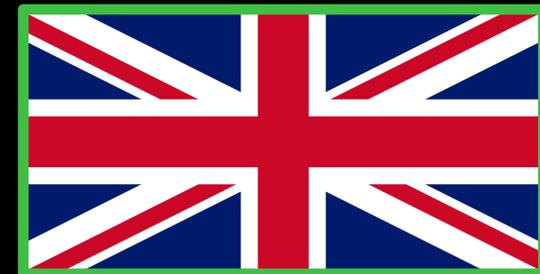


HIDDEN PEARLS

for

HIGH-PERFORMANCE-PERSISTENCE

The Introduction



Who is speaking?



Sven Ruppert

Dev. Advocate – DevSecOps

JFrog Inc

Twitter: @SvenRuppert

Youtube: [DE] - bit.ly/Youtube-Sven

Youtube: [EN] - bit.ly/Outdoor-Nerd

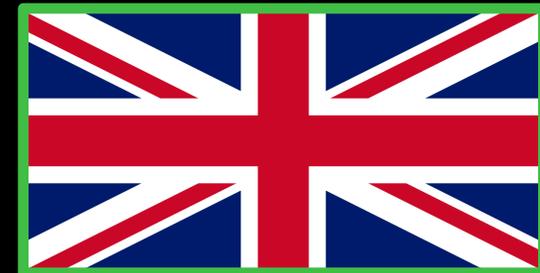


HIDDEN PEARLS

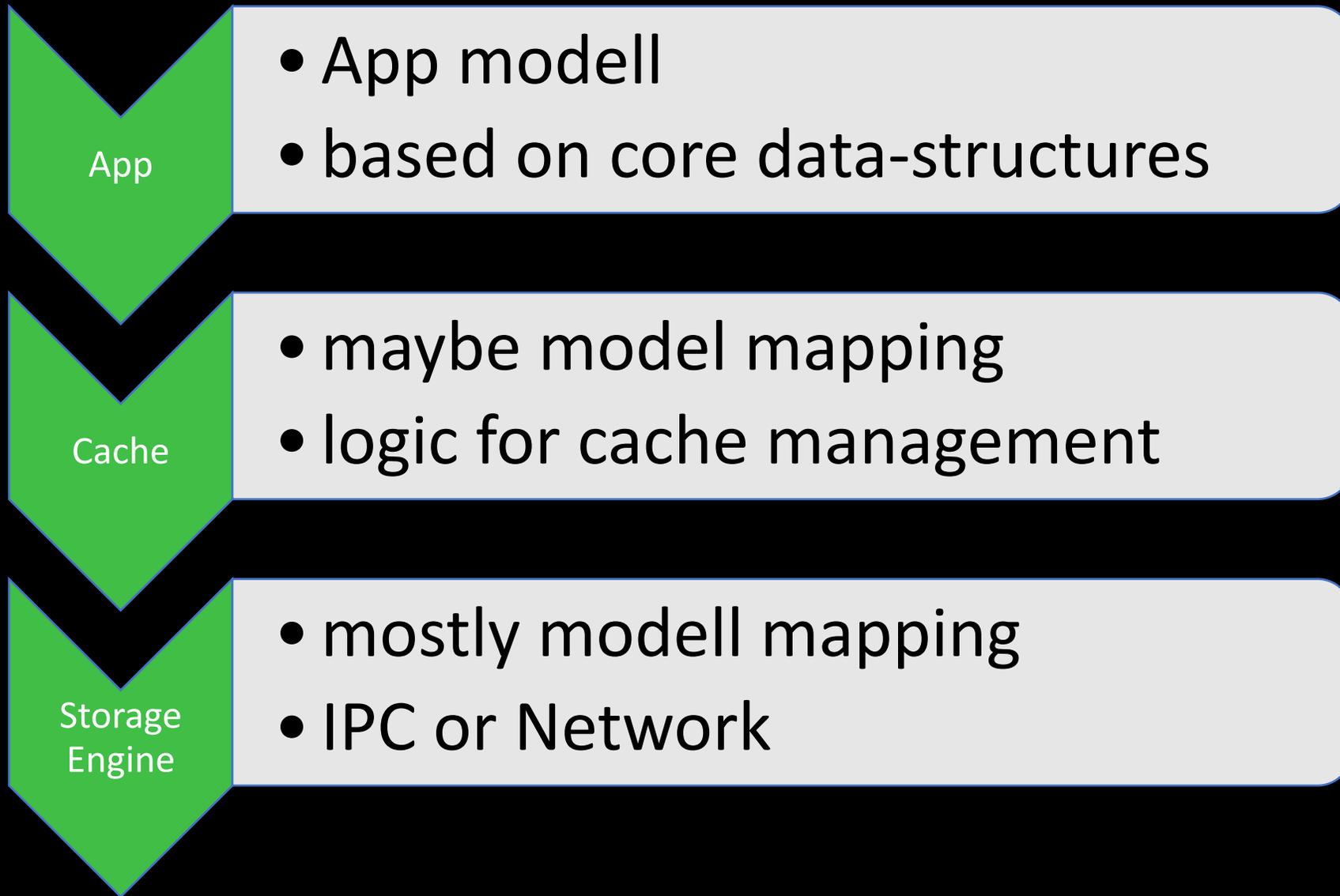
for

HIGH-PERFORMANCE-PERSISTENCE

The Basic Ideas



Persistence BirdEye-View



Basic questions about the data behaviour

Basic questions about the data behaviour

How stable is your data?

Write once, archive it, never touch it

A sliding window of hot data, rest is archived

All data is hot

How to navigate through the data?

SQL , Cypher , Streams , ??

All attributes? Calculated fields? Dynamic Views?

MVCC

Multi-Version Concurrency Control

MVCC

<https://github.com/aidanmorgan/pojo-mvcc>

No dependency – Core Java / Apache Lic

This project contains a simple in-memory Multi-version concurrency control cache for use in Java projects.

In short words: Multi-version concurrency control (MVCC) is a **standard technique for avoiding conflicts between reads and writes of the same object**. POJO-MVCC guarantees that each transaction sees a consistent view of the object by reading non-current data for objects modified by concurrent transactions. MVCC is a fairly common technique in database transaction implementation and is becoming more common in caching implementations.

https://en.wikipedia.org/wiki/Multiversion_concurrency_control

MVCC

In short words:

For every **object** that will be **modified**
the kernel will hold **multiple versions** of the object

every object version will
get different timestamps or transaction numbers for
reads and modifications

positive: non-blocking reads

negative: multiple versions of
an object in memory



HIDDEN PEARLS

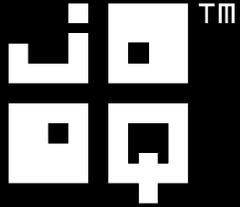
for

HIGH-PERFORMANCE-PERSISTENCE

Some random approaches



Some Random Approaches



JOOQ

```
create
.select(BOOK.TITLE)
.from(BOOK)
.where(BOOK.PUBLISHED_IN
.eq(2011))
.orderBy(BOOK.TITLE)
```

Mapping SQL to Java DSL – SQL-style

Generated from the DB Schema

Some Random Approaches



Streams over **JDBC**
Generated from the DB Schema



JPAStreamer.org

JPAstreamer is released under the [LGPL 2.1 License](#).

Streams over **JPA**

Some Random Approaches



```
jpaStreamer
.stream(Film.class)
.filter(Film$.rating.equal("G"))
.sorted(Film$.length
        .reversed()
        .thenComparing(Film$.title
                        .comparator()))
.skip(10)
.limit(5)
.forEach(System.out::println);
```

Some Random Approaches

Solutions are focussing on convenience only.

Try to get the API Java-like



HIDDEN PEARLS

for

HIGH-PERFORMANCE-PERSISTENCE

Chronicle Maps/Bytes



Chronicle Maps/Bytes

Chronicle **Bytes** contains all the low level memory access wrappers. It is built on Chronicle Core's **direct memory** and **OS system call access**.

<https://github.com/OpenHFT/Chronicle-Bytes>

Chronicle **Map** is a super-fast, in-memory, non-blocking, **key-value** store, designed for low-latency, and/or multi-process applications such as trading and financial market applications.

The size of a Chronicle Map **is not limited by** memory (RAM), but rather by **the available disk capacity**.

<https://github.com/OpenHFT/Chronicle-Map>

Chronicle Bytes

```
<dependency>  
  <groupId>net.openhft</groupId>  
  <artifactId>chronicle-bytes</artifactId>  
  <version>XYZ</version>  
</dependency>
```

```
Bytes<ByteBuffer> bytes = Bytes.elasticHeapByteBuffer(64);  
bytes.writeBoolean(0, true);
```

```
boolean flag = bytes.readBoolean(0);
```

```
MappedBytes mb = MappedBytes.mappedBytes(new File("mapped_file"), 1024);  
mb.appendUtf8("Hello")  
  .append(42.42f);
```

Chronicle Maps

```
<dependency>  
    <groupId>net.openhft</groupId>  
    <artifactId>chronicle-maps</artifactId>  
    <version>XYZ</version>  
</dependency>
```

```
ChronicleMap<LongValue, CharSequence> persistedCountryMap = ChronicleMap  
    .of(LongValue.class, CharSequence.class)  
    .name("value-map")  
    .entries(50)  
    .averageValue("Value-with-Typical-Length")  
    .createPersistedTo(new File("/opt" + "/country-details.dat"));
```



HIDDEN PEARLS

for

HIGH-PERFORMANCE-PERSISTENCE

XODUS



XODUS

Transactional schema-less embedded database used by JetBrains YouTrack and JetBrains Hub.

<https://github.com/JetBrains/xodus>

- Xodus is transactional and fully **ACID-compliant**.
- Xodus is highly concurrent. **Reads** are completely **non-blocking** due to **MVCC** and true snapshot isolation.
- Xodus is schema-less and agile. It does not require schema migrations or refactorings.
- Xodus is **embedded**. It does not require installation or administration.
- Xodus is written **in pure Java and Kotlin**.
- Xodus is free and licensed under Apache 2.0.

XODUS

Snapshot isolation - ONLY

LOG-structured design of XODUS.

All changes are written sequentially to a log. (*.xd – files)

The log is **immutable** !!



persistent functional data structure



Garbage Collector - Needed

XODUS

```
<dependency>  
  <groupId>org.jetbrains.xodus</groupId>  
  <artifactId>xodus-openAPI</artifactId>  
  <version>1.3.232</version>  
</dependency>
```

Environments	org.jetbrains.xodus:xodus-environment:1.3.232
Entity Stores	org.jetbrains.xodus:xodus-entity-store:1.3.232
Virtual File Systems	org.jetbrains.xodus:xodus-vfs:1.3.232

XODUS - Environments

In short, Environment is a transactional **key-value** storage. Store is a named collection of key/value pairs. If a Store is not allowed to contain duplicate keys, then it is a map. Otherwise, it is a multi-map.

```
try (Environment env = Environments.newInstance("/home/me/.myAppData")) {
    env.executeInTransaction(txn -> {
        final Store store = env.openStore("Messages", StoreConfig.WITHOUT_DUPLICATES, txn);
        store.put(txn, StringBinding.stringToEntry("Hello"), StringBinding.stringToEntry("World!"));
    });
}
```

XODUS - Entity Stores

The Entity Stores layer is designed to access data as entities with **attributes** and **links**. Use a transaction to create, modify, read and **query data**. Transactions are quite similar to those on the Environments layer, though the Entity Store API is much richer in terms of querying data.

```
try (PersistentEntityStore entityStore = PersistentEntityStores.newInstance("/home/me/.myAppData")) {
    entityStore.executeInTransaction(txn -> {
        final Entity message = txn.newEntity("Message");
        message.setProperty("hello", "World!");
    });
}
```

XODUS - Virtual File Systems

The VirtualFileSystem lets you deal with data in terms of files, input, and output streams. **VirtualFileSystem** works over an Environment instance:

```
try (Environment env = Environments.newInstance("/home/me/.myAppData")) {
    final VirtualFileSystem vfs = new VirtualFileSystem(env);
    env.executeInTransaction(txn -> {
        final File file = vfs.createFile(txn, "Messages");
        try (DataOutputStream output = new DataOutputStream(vfs.writeFile(txn, file))) {
            output.writeUTF("Hello ");
            output.writeUTF("World!");
        } catch (IOException e) {
            throw new ExodusException(e);
        }
    });
    vfs.shutdown();
}
```



HIDDEN PEARLS

for

HIGH-PERFORMANCE-PERSISTENCE

MapDB



MapDB

MapDB combines embedded database engine and Java collections.
It is free under Apache 2 license.

MapDB is flexible and can be used in many roles:

- Drop-in replacement for **Maps**, **Lists**, **Queues** and other collections.
- **Off-heap** collections not affected by Garbage Collector
- **Multilevel** cache with expiration and disk overflow.
- RDBMs replacement with **transactions**, **MVCC**, incremental backups etc...
- Local data processing and filtering. MapDB has utilities to process huge quantities of data in reasonable time.

<https://github.com/jankotek/mapdb>

Commercial support is offered by the project owner

MapDB

```
<dependency>  
  <groupId>org.mapdb</groupId>  
  <artifactId>mapdb</artifactId>  
  <version>VERSION</version>  
</dependency>
```

```
DB db = DBMaker .fileDB("/some/file")  
  .encryptionEnable("password")  
  .make();
```

```
ConcurrentNavigableMap<Integer,String> map = db.treeMap("collectionName",  
  Serializer.INTEGER, Serializer.STRING)  
  .createOrOpen();
```

```
map.put(1,"one");  
map.put(2,"two");           //map.keySet() is now [1,2] even before commit  
db.commit();               //persist changes into disk  
map.put(3,"three");       //map.keySet() is now [1,2,3]  
db.rollback();            //revert recent changes //map.keySet() is now [1,2]  
db.close();
```

MapDB

```
DB dbDisk = DBMaker .fileDB(file) .make();
```

```
// Big map populated with data expired from cache
```

```
DB dbMemory = DBMaker .memoryDB() .make();
```

```
// fast in-memory collection with limited size
```

```
HTreeMap onDisk = dbDisk .hashMap("onDisk") .create();
```

```
HTreeMap inMemory = dbMemory .hashMap("inMemory")  
    .expireAfterGet(1, TimeUnit.SECONDS) //overflow to `onDisk`  
    .expireOverflow(onDisk) //background expiration  
    .expireExecutor(Executors.newScheduledThreadPool(2))  
    .create();
```

Possible to build **cascaded data-structures** with different attributes.

Combination between **caching** and **persistence**.

Transactions based on **MVCC**

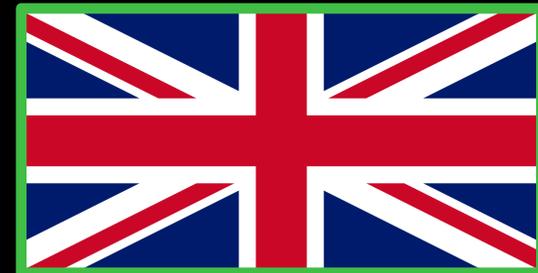


HIDDEN PEARLS

for

HIGH-PERFORMANCE-PERSISTENCE

Microstream



Microstream

<https://microstream.one/java-native-persistence>

Free to use / will be open source soon

MicroStream is a storage technology that stores Java object-graphs natively, which means similar as they are in the RAM, without expensive transformation to any incompatible data structure.

That is the key difference to all database-systems and provides you the following benefits.

- Only 1 data structure (object-graph), only 1 data model (Java classes)
- no mapping
- No loss of performance through mapping
- Simple architecture, super easy to use
- Simplifies and accelerates your entire database development process

Microstream

Free to use / not open source

```
<repository>  
  <id>microstream-releases</id>  
  <url>https://repo.microstream.one/repository/maven-public/</url>  
</repository>
```

```
<dependency>  
  <groupId>one.microstream</groupId>  
  <artifactId>storage.embedded</artifactId>  
  <version>04.00.00-MS-GA</version>  
</dependency>
```

```
<dependency>  
  <groupId>one.microstream</groupId>  
  <artifactId>storage.embedded.configuration</artifactId>  
  <version>04.00.00-MS-GA</version>  
</dependency>
```

Microstream

Free to use / not open source

```
public class HelloWorld {  
    private String value;  
    public String getValue() {  
        return value;  
    }  
    public void setValue(String value) {  
        this.value = value;  
    }  
}
```

```
final HelloWorld value = new HelloWorld();  
value.setValue("HelloWorld");
```

```
final EmbeddedStorageManager storageManager = EmbeddedStorage.start();  
storageManager.setRoot(value);  
storageManager.storeRoot();  
storageManager.shutdown();
```

Microstream

Free to use / not open source

Use a Collection as root – node

Just store your model as it is

No transaction mechanism until now

Loading partial graphs – Virtual Proxies –

```
private Lazy<ArrayList<MyClass>> myClasses = Lazy.Reference(new ArrayList<>());
```

Cycles in the graph are not a problem

No special Inheritance

<https://dzone.com/articles/high-performance-persistence-with-microstream-part>

Microstream

Free to use / not open source

Use a Collection as root – node

Just store your model as it is

No transaction mechanism until now

Loading partial graphs – Virtual Proxies –

```
private Lazy<ArrayList<MyClass>> myClasses = Lazy.Reference(new ArrayList<>());
```

Cycles in the graph are not a problem

No special Inheritance

<https://dzone.com/articles/high-performance-persistence-with-microstream-part>

Microstream and MVCC

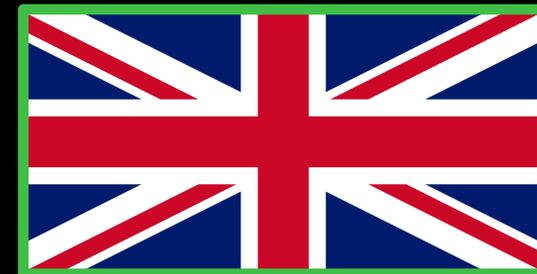


HIDDEN PEARLS

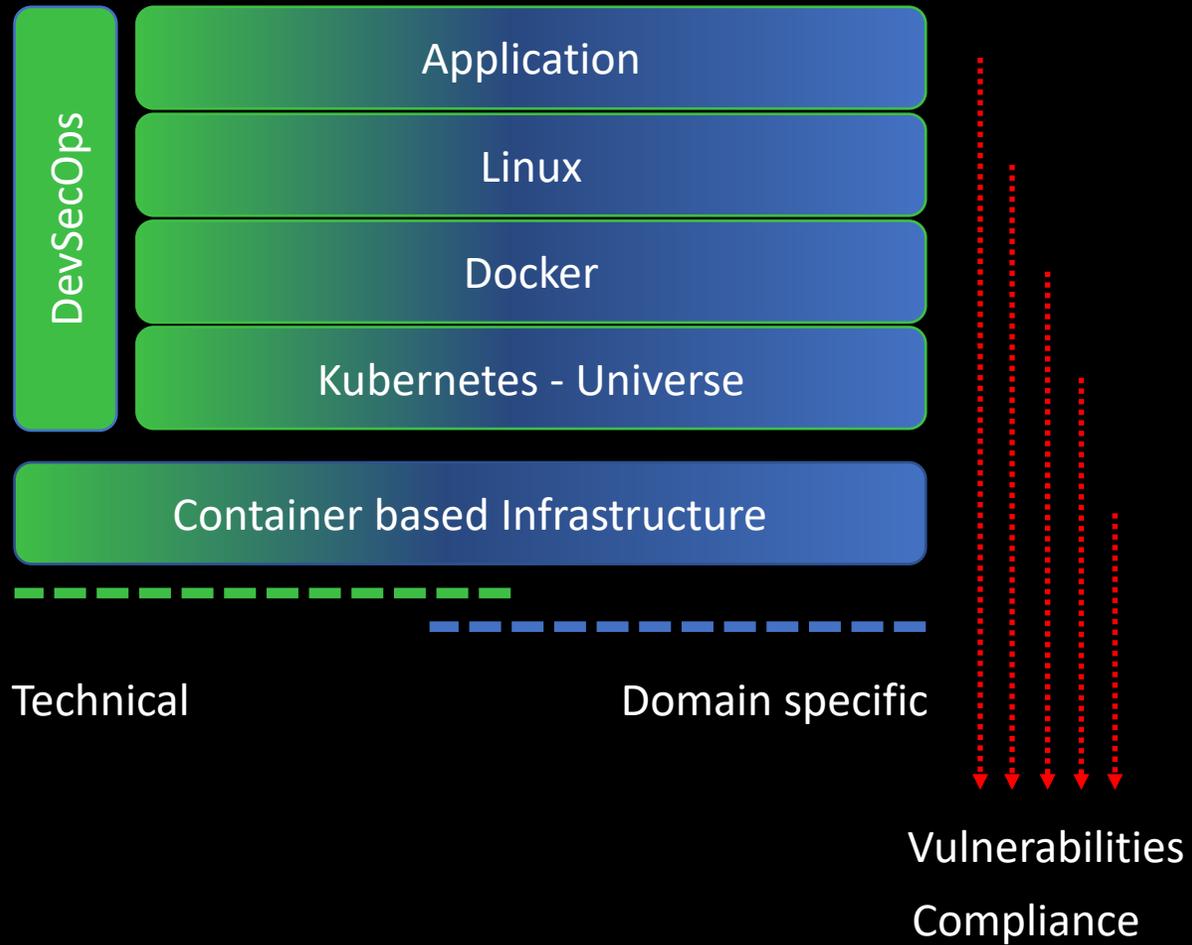
for

HIGH-PERFORMANCE-PERSISTENCE

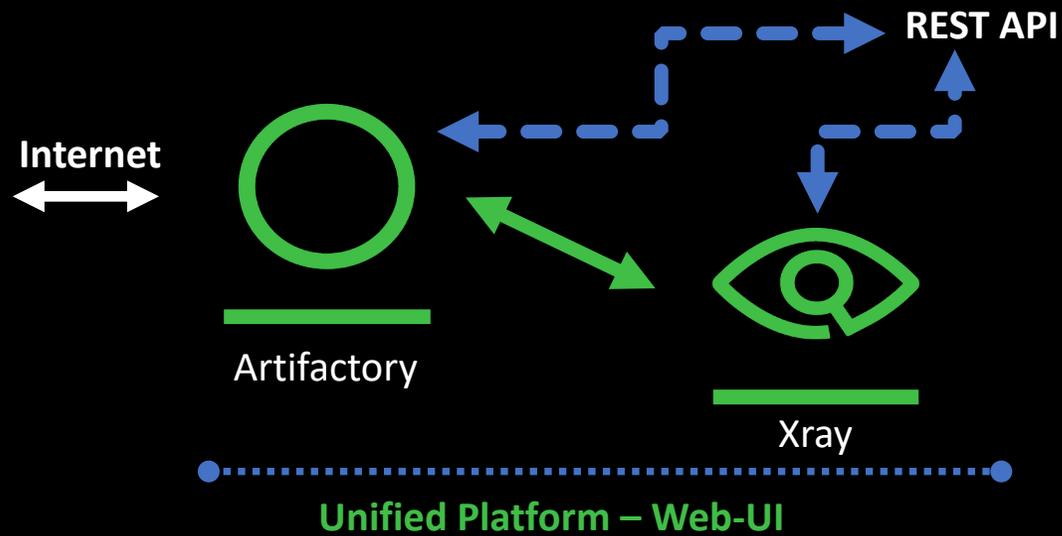
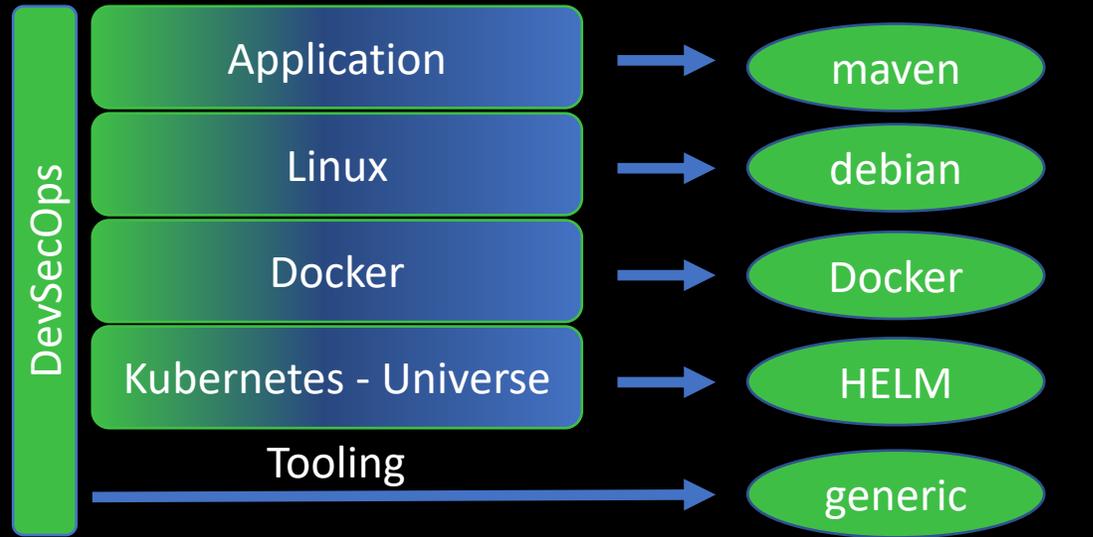
DevSecOps / Security



Persistence and Dev(Sec)Ops



Persistence and Dev(Sec)Ops



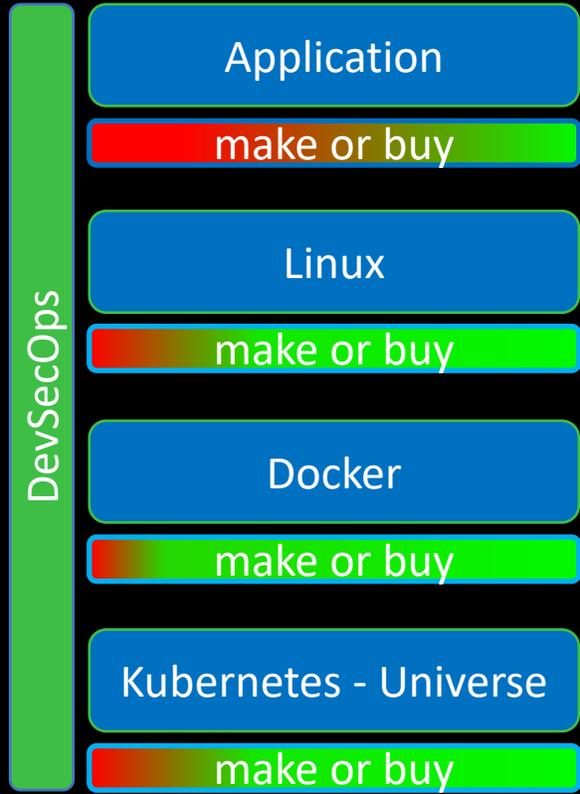
Low-hanging Fruits - Dependencies

Time to market

Make or Buy

Don't re-invent the wheel

Low-hanging Fruits - Dependencies



make

buy

→ Dependency

Low-hanging Fruits - Dependencies

Compliance

One time effort to define allowed Lic

Machine is doing the job

Vulnerabilities

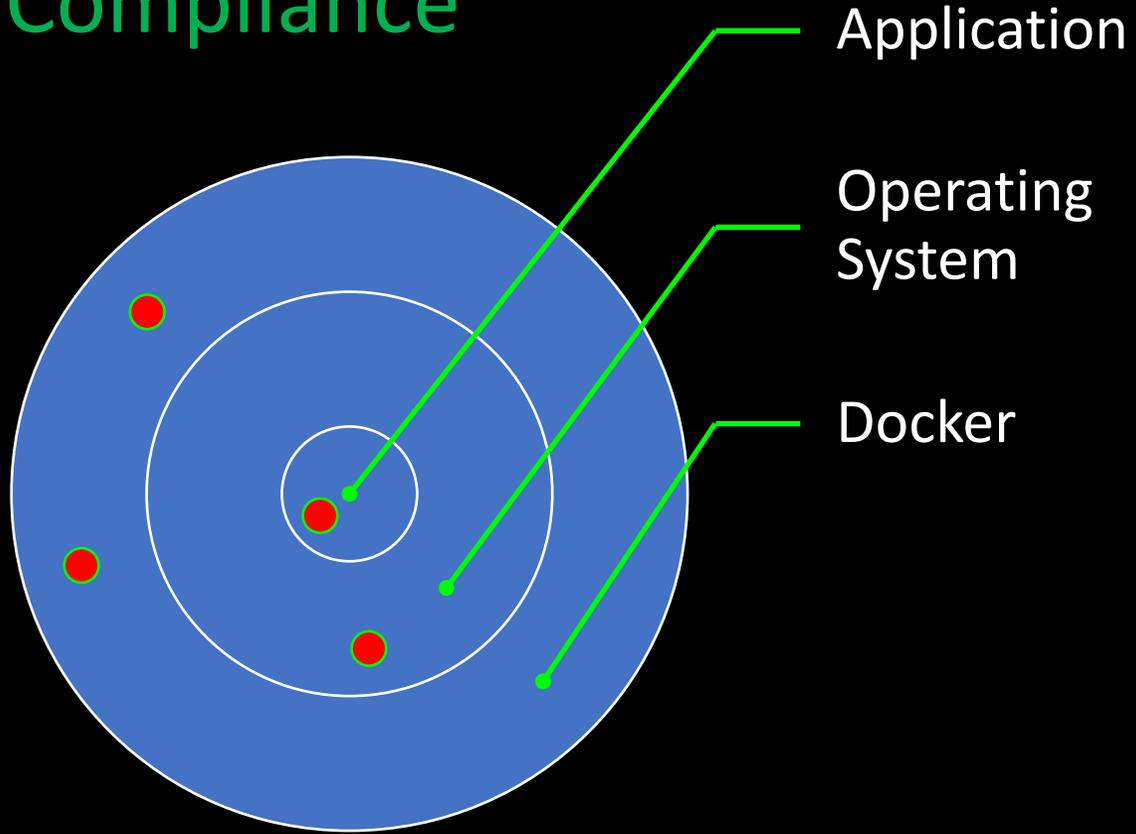
recurrent effort

Machine is scanning

Human is deciding

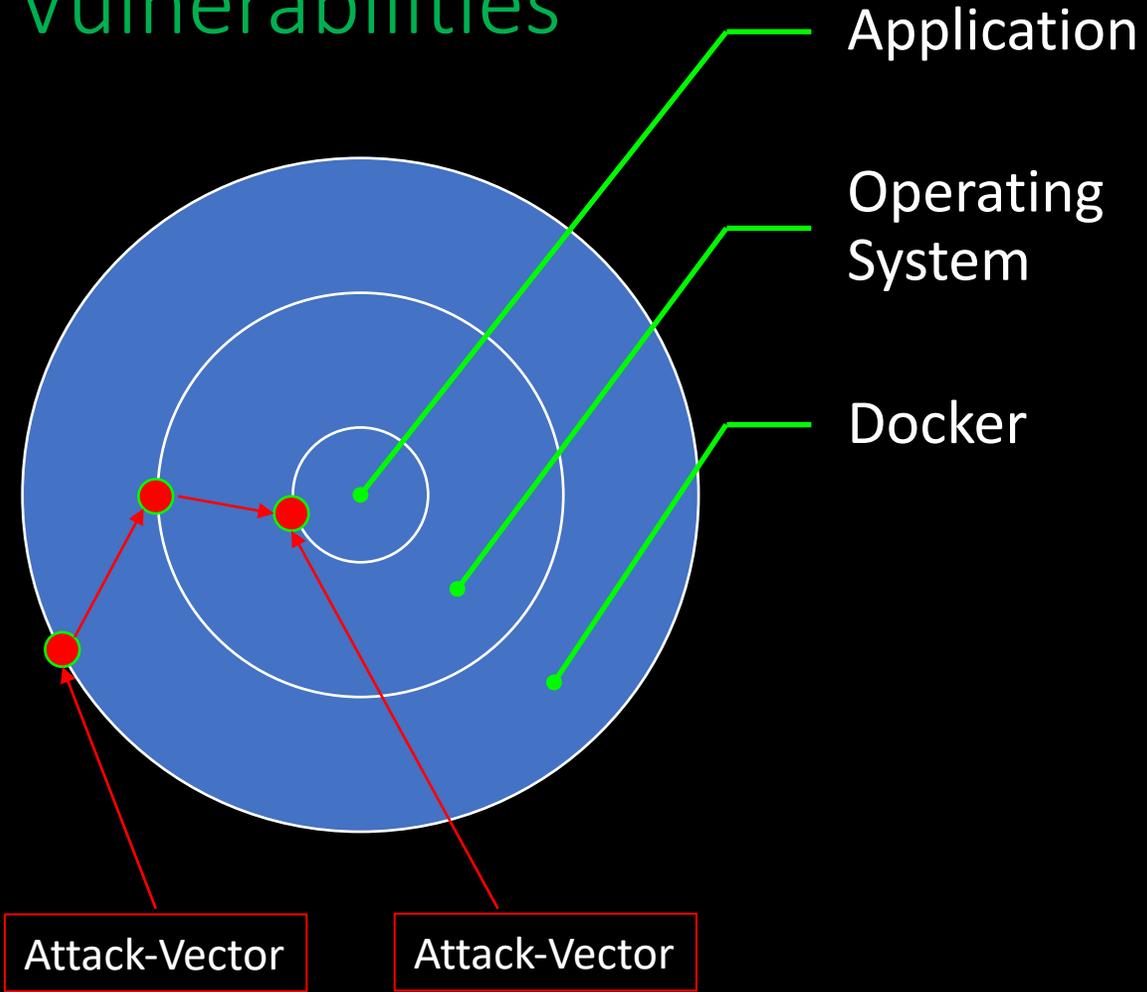
Low-hanging Fruits - Dependencies

Compliance



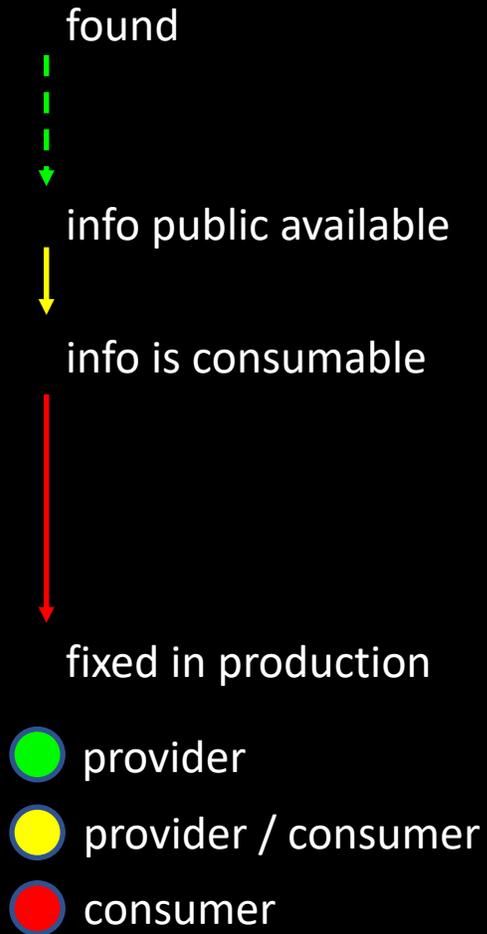
Low-hanging Fruits - Dependencies

Vulnerabilities



Low-hanging Fruits - Dependencies

Vulnerabilities



Low-hanging Fruits - Dependencies

Vulnerabilities / Compliance

Good Test-Coverage is your safety-belt

Optimize your deployment times

Dependencies are the biggest part



Dependency Management
has the highest impact

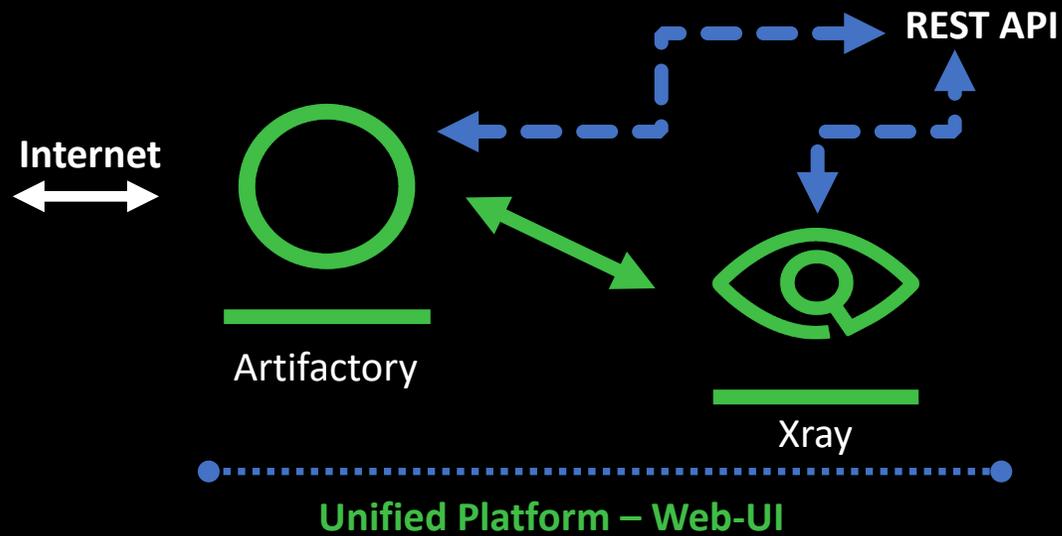
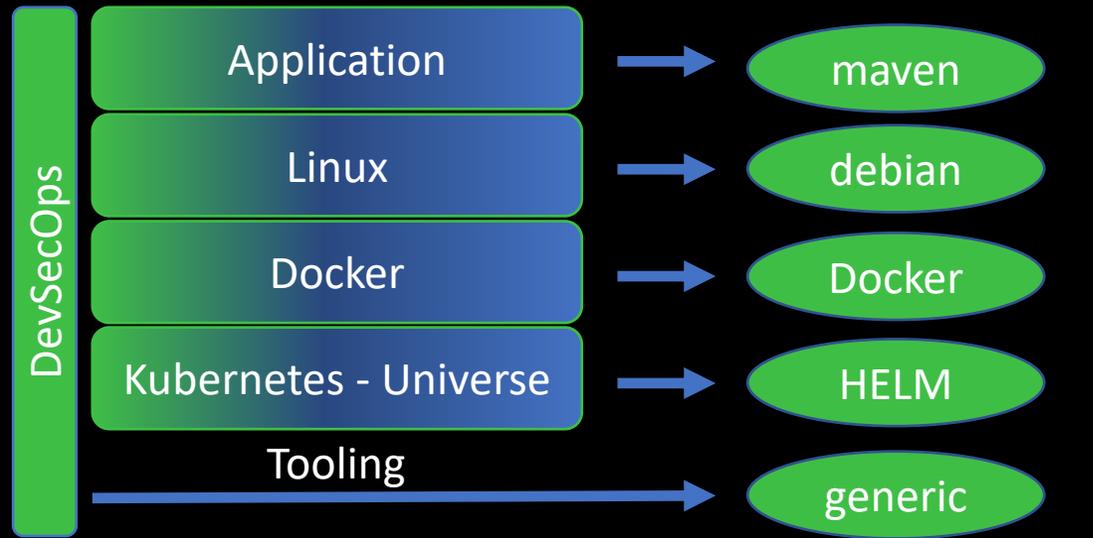
MUTATION TESTING



START HUNTING THE BUGS



Low-hanging Fruits - Dependencies



Persistence and Dev(Sec)Ops

Random thoughts

- Smaller amount of components – mostly easier to handle
- Shorter RamUp / RampDown times for TDD
- Datamigration – depends on the system itself
- DataDrivenTesting – just a few binaries hosted in a repo
 - Generic Repository in Artifactory - immutable

Compliance / Security Issues

- Lic check of all components
- Security check of all dependencies , including transitive ones
- As Early as possible #ShiftLeft

Persistence and Dev(Sec)Ops

Only one of the discussed systems have a **Security Issue** right now

Based on the missing components the general amount of attack vectors are smaller

Complete stack is managed via maven

- Easy to integrate into CI environments
- IDE Support via CodeCompletion

Tooling is **not complete**:

- DataNavigation / AdHoc query
- Backup / Restore
- ...

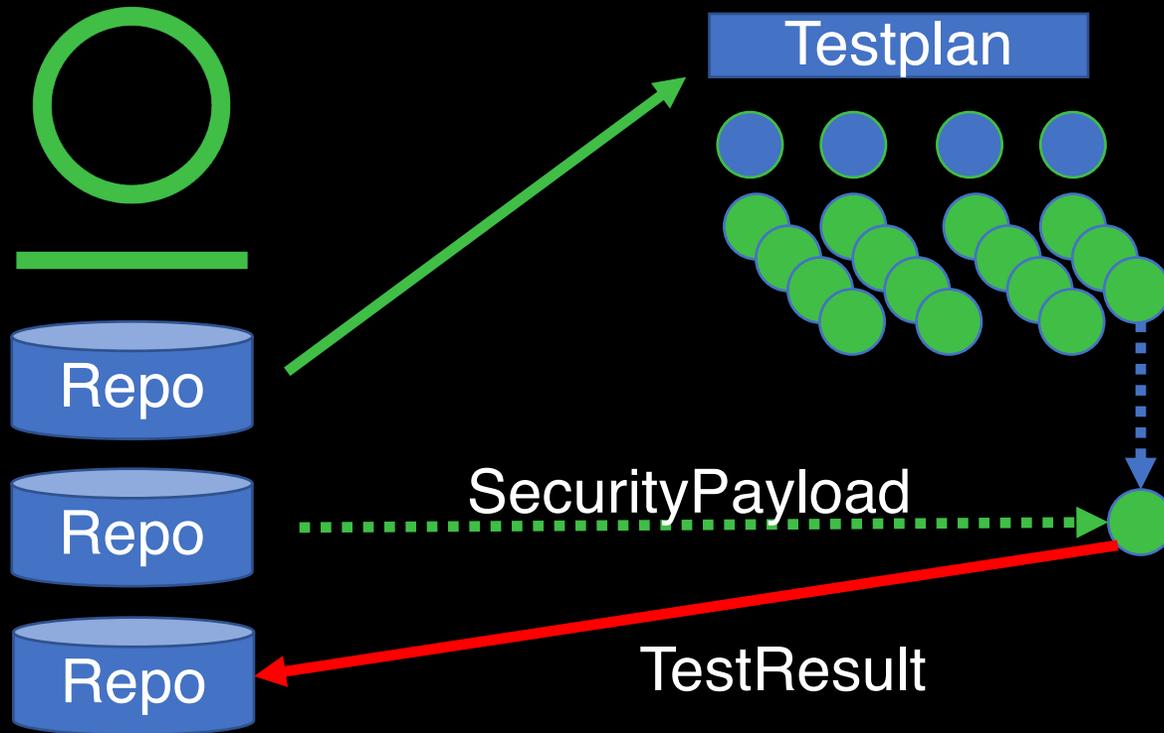
SecurityPayload Generator Injection

Load binary data from Artifactory – Generic Repo

SecurityPayload Generator Injection

Store a TestPlan with the BuildInfo

Store a TestResults with the BuildInfo



JFROG HOWTO

FREE-TIER ACTIVATION



DEVOPS
FAST FORWARD



@SVENRUPPERT

JFROG HOWTO

VULNERABILITIES REPORT INTRO



@SVENRUPPERT

Additional Content

- <https://microstream.one/>
- <https://github.com/JetBrains/xodus>
- <https://github.com/jankotek/mapdb>
- <https://github.com/OpenHFT/Chronicle-Bytes>
- <https://github.com/OpenHFT/Chronicle-Map>

Newsletter – Sven private for the Community

<https://mailings.sven-ruppert.de/>

With this newsletter, you have access to the **slides of my videos**, and you are one of the **first to be informed** about my **new videos, articles** and **blog posts**.

You will also always receive the latest information about where and when I will shortly give **online lectures, meetups** and **free webinars** so that you can **secure a place for yourself**.



Sven Ruppert

Dev. Advocate – DevSecOps

JFrog Inc

Twitter: [@SvenRuppert](#)

Youtube: [DE] - bit.ly/Youtube-Sven

Youtube: [EN] - bit.ly/Outdoor-Nerd

THANK YOU!