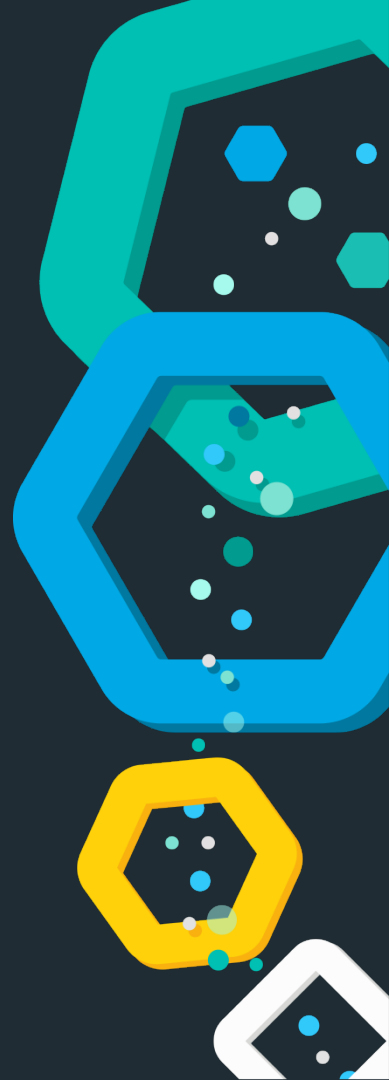


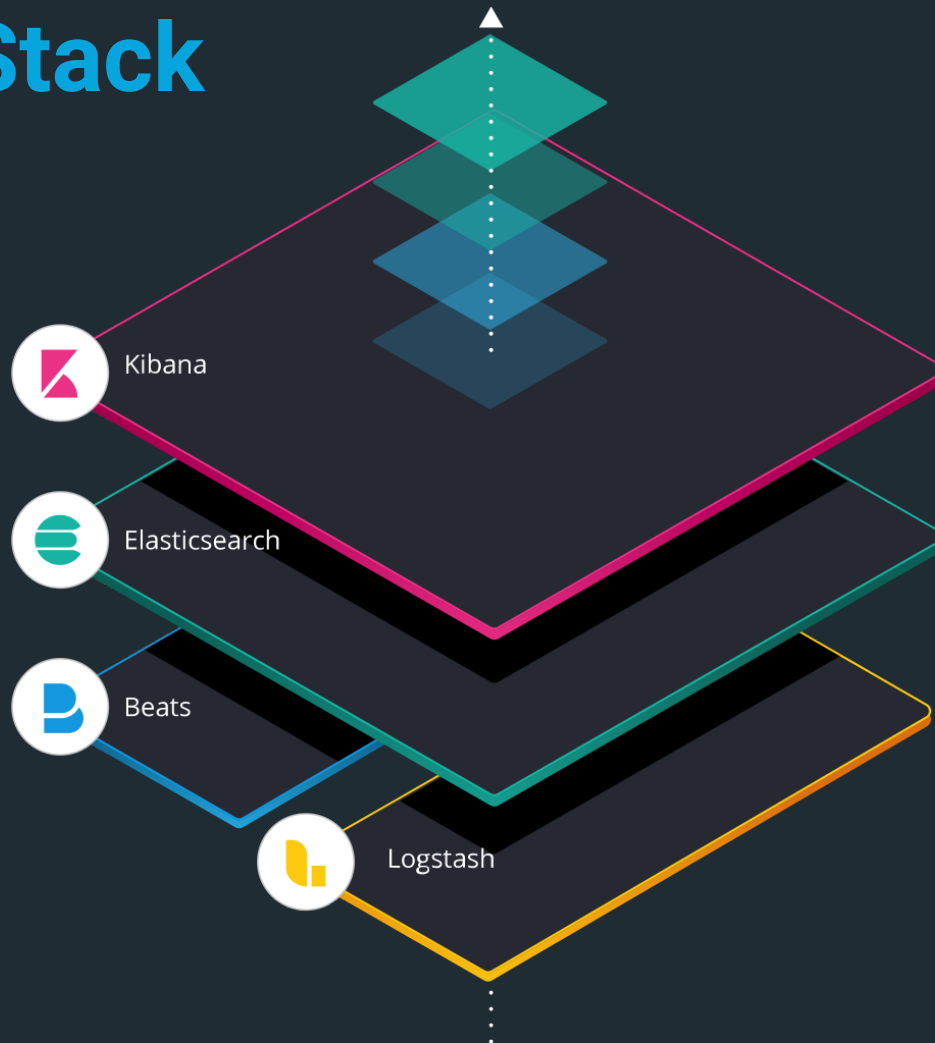
# Elasticsearch

## Securing a search engine while maintaining usability

Alexander Reelsen  
@spinscale  
alex@elastic.co



# Elastic Stack



# Elasticsearch in 10 seconds

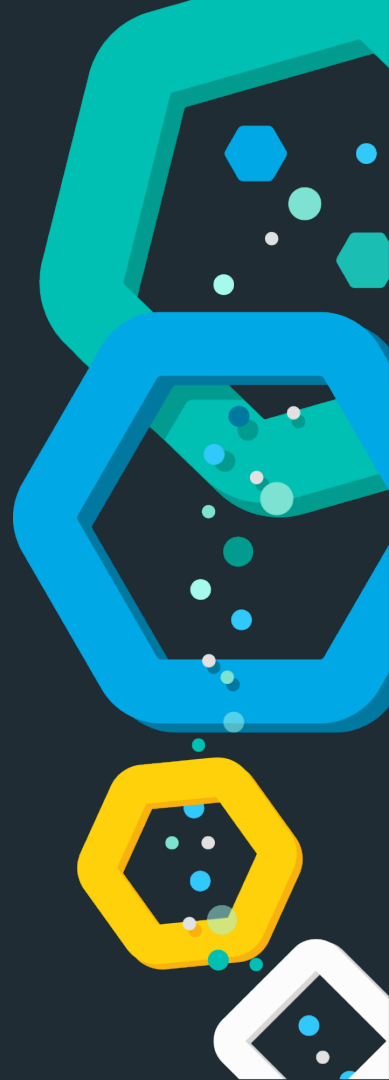
- 🌿 Search Engine (FTS, Analytics, Geo), near real-time
- 🌿 Distributed, scalable, highly available, resilient
- 🌿 Interface: HTTP & JSON
- 🌿 Centrepiece of the Elastic Stack (Kibana, Logstash, Beats, APM, ML, App Search, Enterprise Search)
- 🌿 Uneducated conservative guess: Tens of thousands of clusters worldwide, hundreds of thousands of instances

# Agenda

- 🌿 Security: Feature or non-functional requirement?
- 🌿 Security Manager
- 🌿 Production Mode vs. Development Mode
- 🌿 Plugins
- 🌿 Scripting language: Painless

# Security

Feature or non-functional requirement?



# Security as a non-functional requirement

- 🌸 Software has to be secure! O RLY?
- 🌸 Defensive programming
- 🌸 Do not persist specific data (PCI DSS)
- 🌸 Not exploitable (pro tip: not gonna happen)
- 🌸 No unintended resource access (directory traversal)
- 🌸 Least privilege principle
- 🌸 Reduced impact surface (DoS)

## Security

### Dishwasher has directory traversal bug

Thanks a Miele-on for making everything dangerous, Internet of Things firmware slackers

By [Richard Chirgwin](#) 26 Mar 2017 at 23:08

192 [SHARE](#) ▼



Don't say you weren't warned: Miele went full Internet-of-Things with a network-connected dishwasher, gave it a web server, and now finds itself on the wrong end of a security bug report – *and* it's accused of ignoring the warning.

The utterly predictable [vulnerability advisory](#) on the Full Disclosure mailing list details [CVE-2017-7240](#) – aka "Miele Professional PG 8528 - Web Server Directory Traversal." This is the builtin web server that's used to remotely control the glassware-cleaning machine from a browser.

# Security as a feature

- 🌸 Authentication
- 🌸 Authorization (LDAP, users, PKI)
- 🌸 TLS transport encryption
- 🌸 Audit logging
- 🌸 SSO/SAML/Kerberos



# Security or safety or resiliency?

- 🍷 Integrity checks
- 🍷 Preventing OOMEs
- 🍷 Prevent deep pagination
- 🍷 Do not expose credentials in cluster state/REST APIs
- 🍷 Stop writing data before running out of disk space
- 🍷 Unable to call `System.exit`



„[T]HERE ARE **KNOWN KNOWNS**; THERE ARE THINGS WE KNOW WE KNOW. WE ALSO KNOW THERE ARE **KNOWN UNKNOWN**S; THAT IS TO SAY WE KNOW THERE ARE SOME THINGS WE DO NOT KNOW. BUT THERE ARE ALSO **UNKNOWN UNKNOWN**S – THERE ARE THINGS WE DO NOT KNOW WE DON'T KNOW.“

Donald Rumsfeld, ~~former secretary of defense~~, IT Security Expert

„[T]HERE ARE **KNOWN KNOWNS**; THERE ARE THINGS WE KNOW WE KNOW. WE ALSO KNOW THERE ARE **KNOWN UNKNOWN**S; THAT IS TO SAY WE KNOW THERE ARE SOME THINGS WE DO NOT KNOW. BUT THERE ARE ALSO **UNKNOWN UNKNOWN**S – THERE ARE THINGS WE DO NOT KNOW WE DON'T KNOW.“

Donald Rumsfeld, ~~former secretary of defense~~, IT Security Expert

„[T]HERE ARE **KNOWN KNOWNS**; THERE ARE THINGS WE KNOW WE KNOW. WE ALSO KNOW THERE ARE **KNOWN UNKNOWN**S; THAT IS TO SAY WE KNOW THERE ARE SOME THINGS WE DO NOT KNOW. BUT THERE ARE ALSO **UNKNOWN UNKNOWN**S — THERE ARE THINGS WE DO NOT KNOW WE DON'T KNOW.“

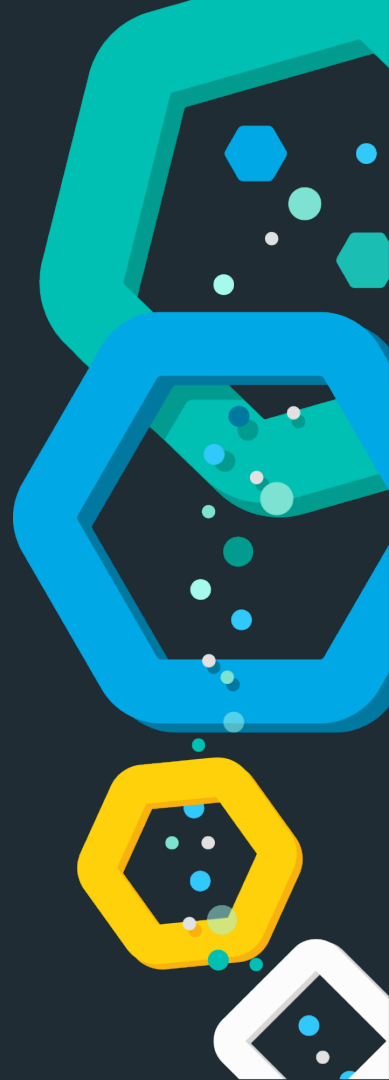
Donald Rumsfeld, ~~former secretary of defense~~, IT Security Expert

„[T]HERE ARE **KNOWN KNOWNS**; THERE ARE THINGS WE KNOW WE KNOW. WE ALSO KNOW THERE ARE **KNOWN UNKNOWN**S; THAT IS TO SAY WE KNOW THERE ARE SOME THINGS WE DO NOT KNOW. BUT THERE ARE ALSO **UNKNOWN UNKNOWN**S – THERE ARE THINGS WE DO NOT KNOW WE DON'T KNOW.“

Donald Rumsfeld, ~~former secretary of defense~~, IT Security Expert

# Security Manager

Have you ever called `System.setSecurityManager()`?



# What is a sandbox?

Your code

```
connect 192.168.1.1:9300
```



```
write /var/log/elasticsearch.log
```



```
unlink /var/lib/elasticsearch/...
```



# What is a sandbox?

Your code

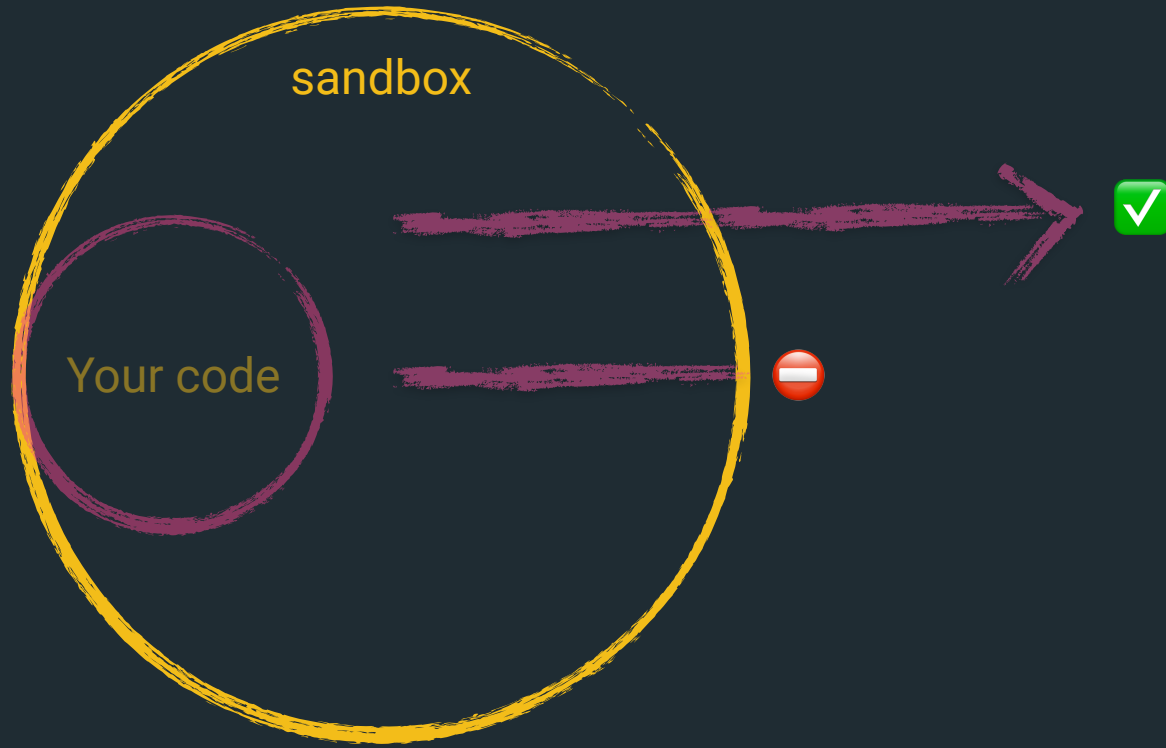
`open /etc/passwd`

`connect bitcoin-miner.foo.bar`

`unlink /var/lib/elasticsearch`



# What is a sandbox?





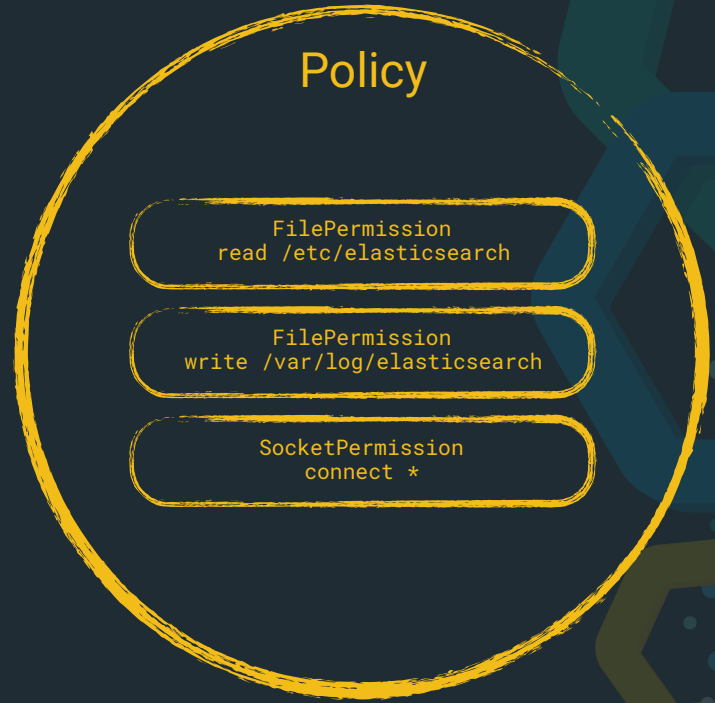
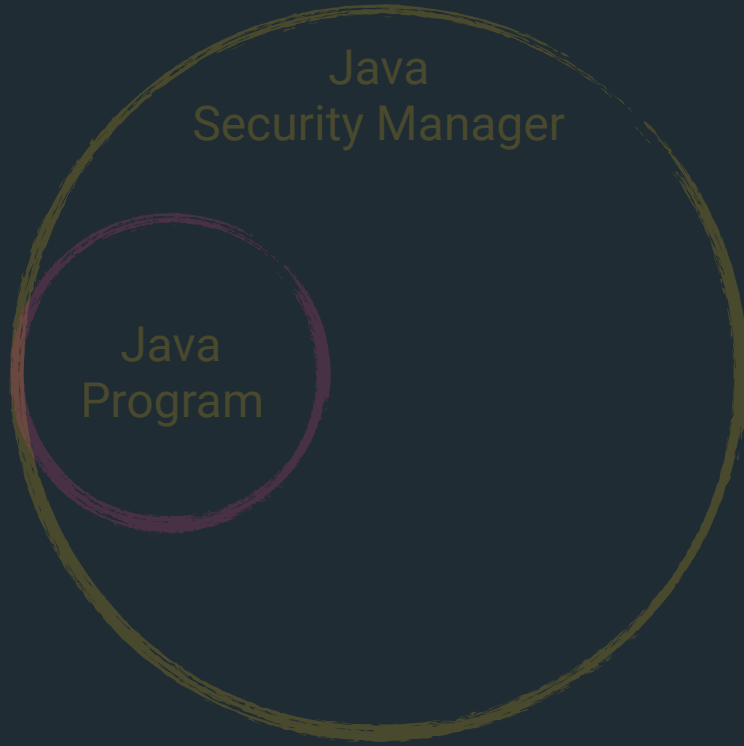
# Introduction

- 🍷 Sandbox your java application
- 🍷 Prevent certain calls by your application
- 🍷 Policy file grants permissions
  - 🍷 `FilePermission` (read, write)
  - 🍷 `SocketPermission` (connect, listen, accept)
  - 🍷 `URLPermission`, `PropertyPermission`, ...

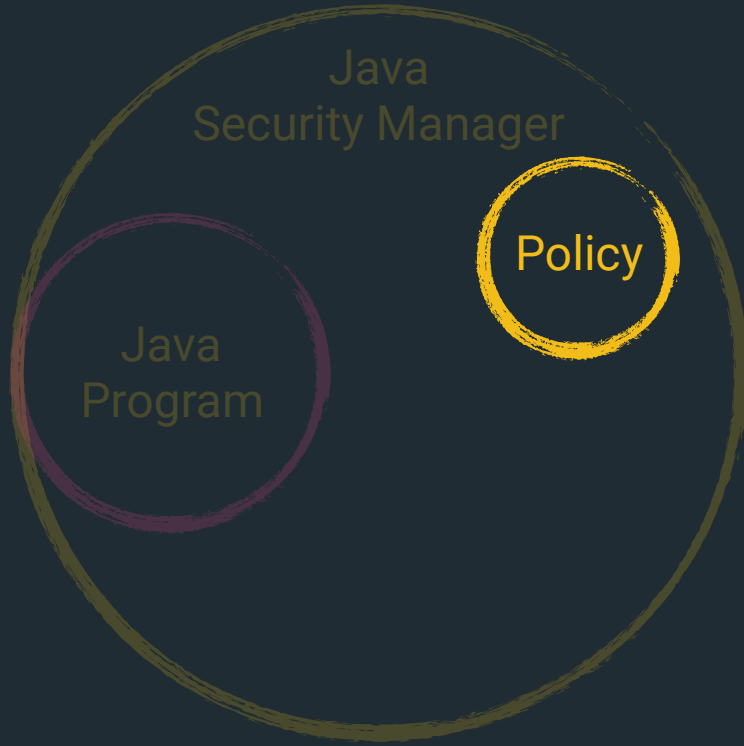
# Java Security Manager



# Java Security Manager



# Java Security Manager



# Introduction

```
/**
 * Tests whether the file or directory denoted by this abstract pathname
 * exists.
 *
 * @return true if and only if the file or directory denoted
 * by this abstract pathname exists; false otherwise
 *
 * @throws SecurityException
 * If a security manager exists and its @link
 * java.lang.SecurityManager#checkRead(java.lang.String)
 * method denies read access to the file or directory
 */
public boolean exists() {
    SecurityManager security = System.getSecurityManager();
    if (security != null) {
        security.checkRead(path);
    }
    if (isInvalid()) {
        return false;
    }
    return ((fs.getBooleanAttributes(f: this) & FileSystem.BA_EXISTS) != 0);
}
```

# Introduction

```
/**
 * Throws a SecurityException if the
 * calling thread is not allowed to read the file specified by the
 * string argument.
 * <p>
 * This method calls checkPermission with the
 * FilePermission(file,"read") permission.
 * <p>
 * If you override this method, then you should make a call to
 * super.checkRead
 * at the point the overridden method would normally throw an
 * exception.
 *
 * @param file the system-dependent file name.
 * @exception SecurityException if the calling thread does not have
 * permission to access the specified file.
 * @exception NullPointerException if the file argument is
 * null.
 * @see #checkPermission(java.security.Permission) checkPermission
 */
public void checkRead(String file) {
    checkPermission(new FilePermission(file,
        SecurityConstants.FILE_READ_ACTION));
}
```



# DEMO



# OHAI JLS

## 17.5.3. Subsequent Modification of `final` Fields

In some cases, such as deserialization, the system will need to change the `final` fields of an object after construction. `final` fields can be changed via reflection and other implementation-dependent means. The only pattern in which this has reasonable semantics is one in which an object is constructed and then the `final` fields of the object are updated. The object should not be made visible to other threads, nor should the `final` fields be read, until all updates to the `final` fields of the object are complete. Freezes of a `final` field occur both at the end of the constructor in which the `final` field is set, and immediately after each modification of a `final` field via reflection or other special mechanism.

<https://docs.oracle.com/javase/specs/jls/se11/html/jls-17.html#jls-17.5.3>

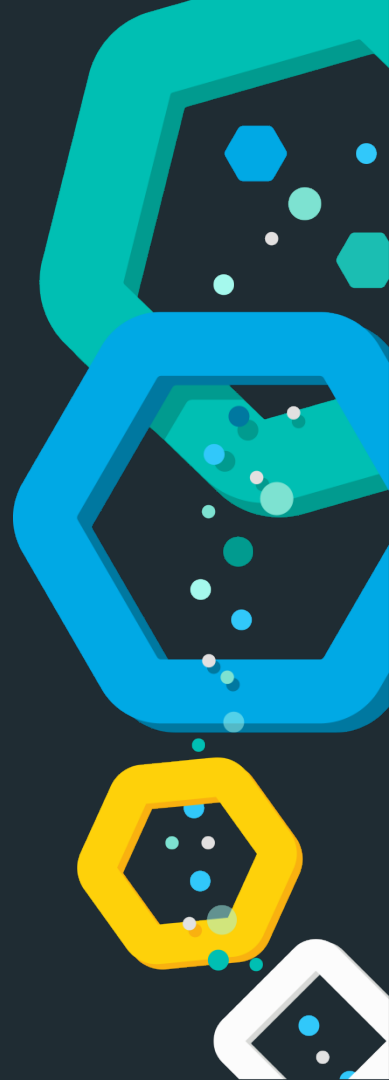


# Drawbacks

- 🌀 Hardcoded policies before startup
- 🌀 DNS lookups are cached forever by default
- 🌀 Forces you to think about dependencies!
- 🌀 Many libraries are not even tested with the security manager, unknown code paths may be executed
- 🌀 No OOM protection! No stack overflow protection!
- 🌀 Granularity
- 🌀 No protection against java agents

# Reducing impact

Bad things have less bad results



# Reducing impact

- 🌿 Elasticsearch integration of the Java Security Manager
- 🌿 Least privilege principle
- 🌿 Do not run as root
- 🌿 No chance of forking a process
- 🌿 Do not expose sensitive settings

# Security Manager in Elasticsearch

- ❁ Initialization required before starting security manager
- ❁ Elasticsearch needs to read its configuration file first to find out about the file paths
- ❁ Native code needs to be executed first
- ❁ Solution: Start with empty security manager, bootstrap, apply secure security manager

# Elasticsearch startup

JVM Startup



time



# Elasticsearch startup

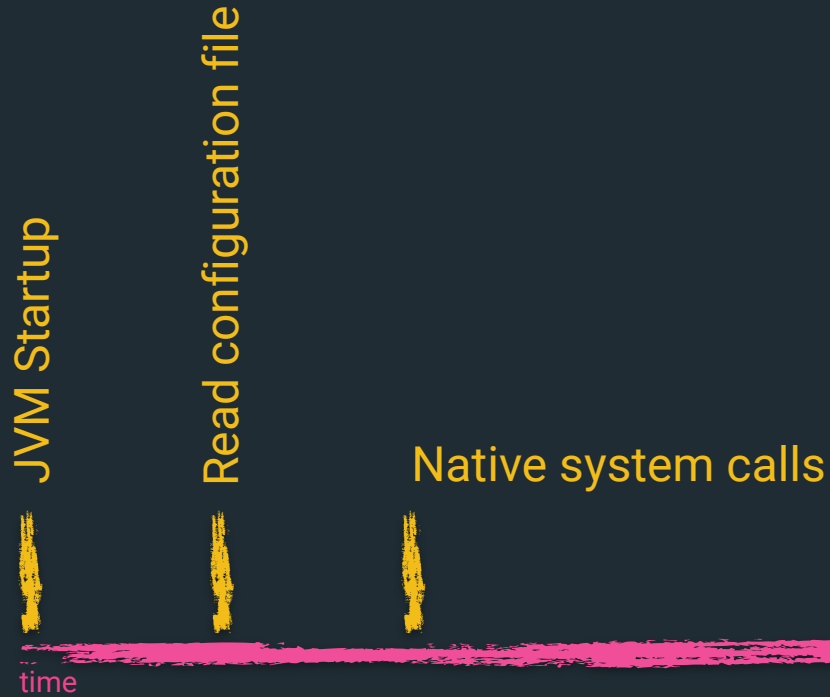
JVM Startup

Read configuration file

time



# Elasticsearch startup

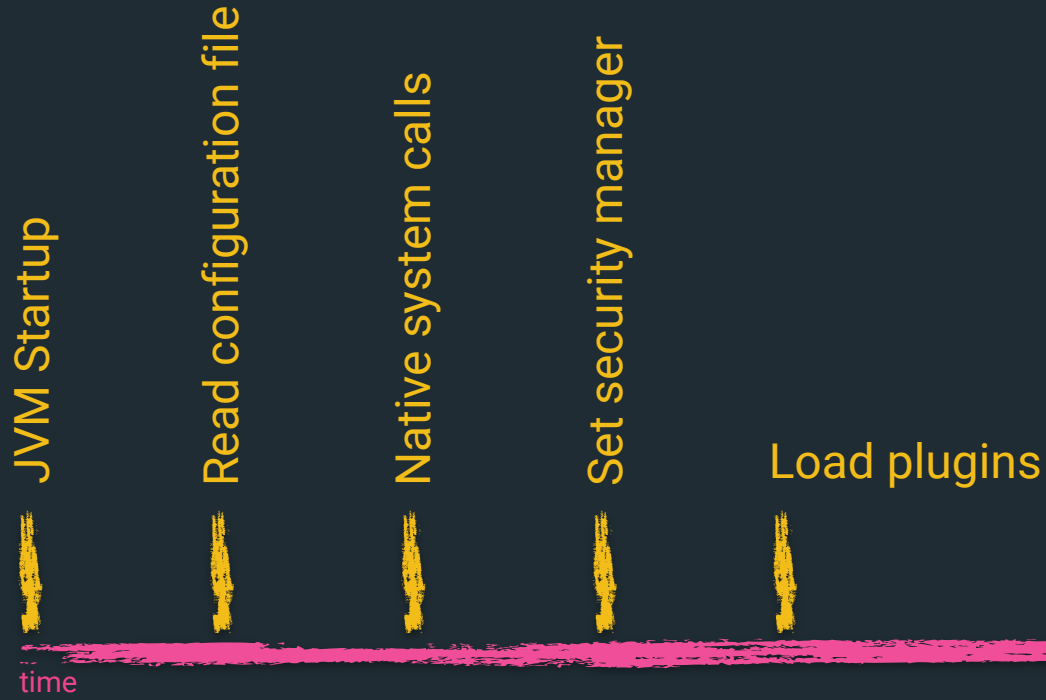


# Elasticsearch startup

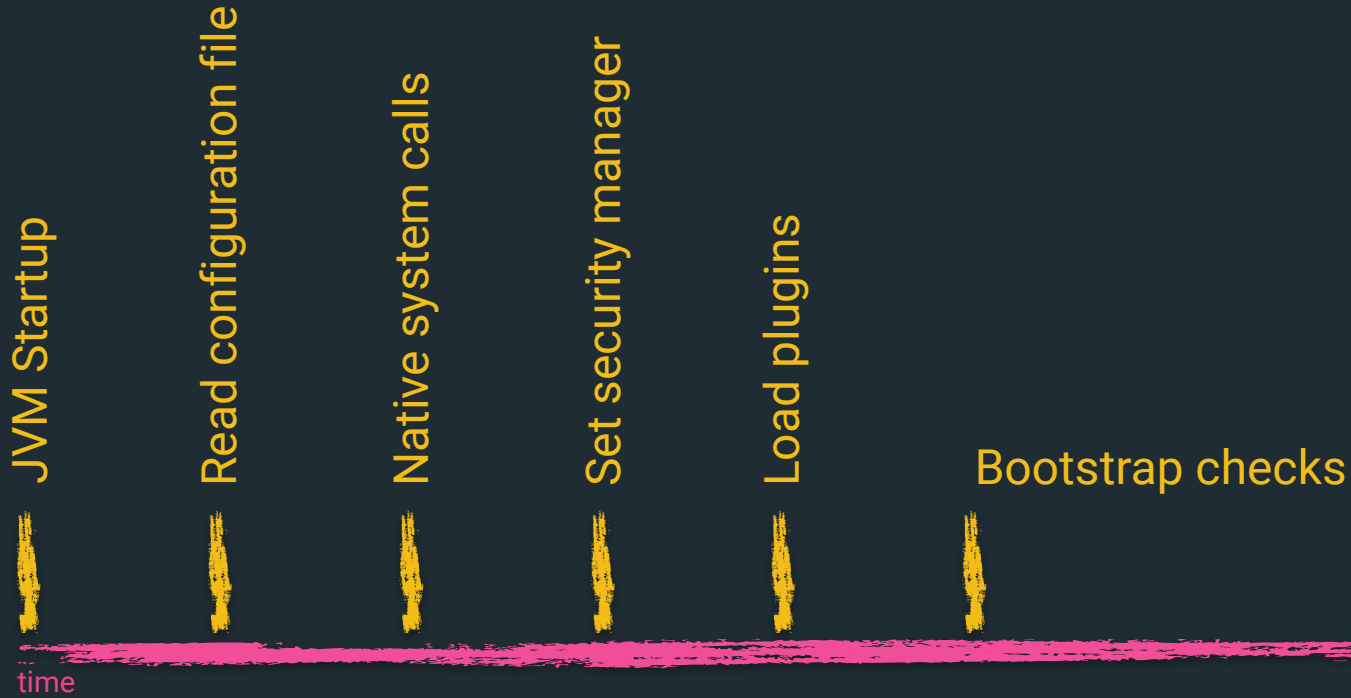




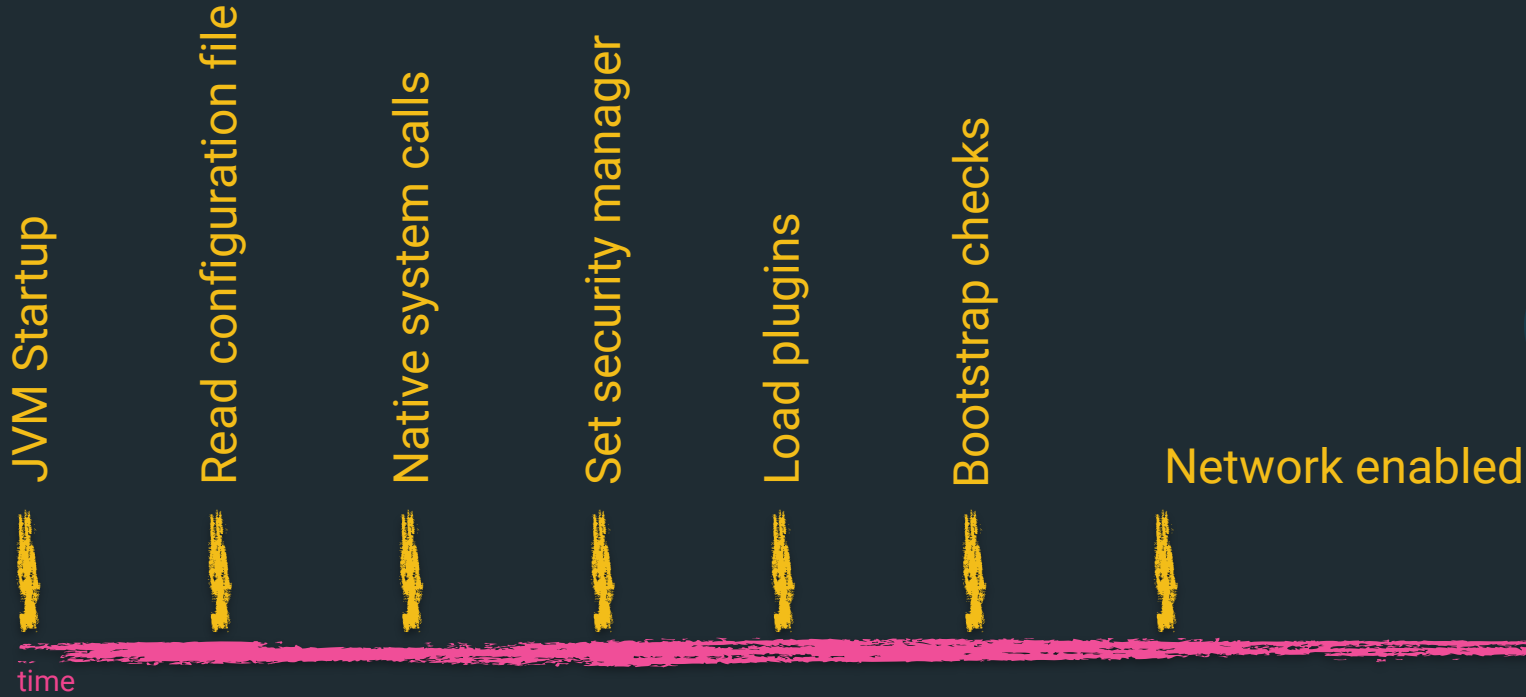
# Elasticsearch startup



# Elasticsearch startup



# Elasticsearch startup



# Elasticsearch startup



# Security Manager in Elasticsearch

- ❁ Special security manager is used
- ❁ Does not set `exitVM` permissions, only a few special classes are allowed to call
- ❁ Thread & ThreadGroup security is enforced
- ❁ Also **SpecialPermission** was added, a special marker permission to prevent elevation by scripts

# Security Manager in Elasticsearch

- ❁ **ESPolicy** allows for loading from files plus dynamic configuration (from the ES configuration file)
- ❁ Bootstrap check for **`java.security.AllPermission`**

# #noroot

there is no reason to run code as root!



# Do not run as root

JVM Startup

Read configuration file

**Native system calls**

Set security manager

Load plugins

Bootstrap checks

Network enabled

time





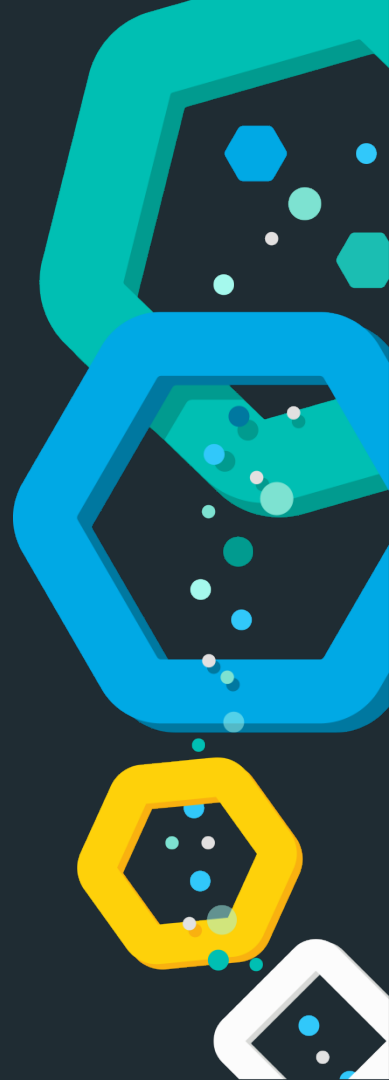
# Do not run as root

```
/** Returns true if user is root, false if not, or if we don't know */
static boolean definitelyRunningAsRoot() {
    if (Constants.WINDOWS) {
        return false; // don't know
    }
    try {
        return JNACLibrary.geteuid() == 0;
    } catch (UnsatisfiedLinkError e) {
        // this will have already been logged by Kernel32Library, no need to repeat it
        return false;
    }
}
```

```
// check if the user is running as root, and bail
if (Natives.definitelyRunningAsRoot()) {
    throw new RuntimeException("can not run elasticsearch as root");
}
```

# seccomp

... or how I loved to abort system calls



# Seccomp - prevent process forks



# Seccomp - prevent process forks

- ❁ Security manager could fail
- ❁ Elasticsearch should still not be able to fork processes
- ❁ One way transition to tell the operating system to deny `execve`, `fork`, `vfork`, `execveat` system calls
- ❁ Works on Linux, Windows, Solaris, BSD, osx

# Seccomp - prevent process forks

```
// BPF installed to check arch, limit, then syscall.
// See https://www.kernel.org/doc/Documentation/prctl/seccomp\_filter.txt for details.
SockFilter insns[] = {
    /* 1 */ BPF_STMT( code: BPF_LD + BPF_W + BPF_ABS, SECCOMP_DATA_ARCH_OFFSET), //
    /* 2 */ BPF_JUMP( code: BPF_JMP + BPF_JEQ + BPF_K, arch.audit, jt: 0, jf: 7), // if (arch != audit) goto fail;
    /* 3 */ BPF_STMT( code: BPF_LD + BPF_W + BPF_ABS, SECCOMP_DATA_NR_OFFSET), //
    /* 4 */ BPF_JUMP( code: BPF_JMP + BPF_JGT + BPF_K, arch.limit, jt: 5, jf: 0), // if (syscall > LIMIT) goto fail;
    /* 5 */ BPF_JUMP( code: BPF_JMP + BPF_JEQ + BPF_K, arch.fork, jt: 4, jf: 0), // if (syscall == FORK) goto fail;
    /* 6 */ BPF_JUMP( code: BPF_JMP + BPF_JEQ + BPF_K, arch.vfork, jt: 3, jf: 0), // if (syscall == VFORK) goto fail;
    /* 7 */ BPF_JUMP( code: BPF_JMP + BPF_JEQ + BPF_K, arch.execve, jt: 2, jf: 0), // if (syscall == EXECVE) goto fail;
    /* 8 */ BPF_JUMP( code: BPF_JMP + BPF_JEQ + BPF_K, arch.execveat, jt: 1, jf: 0), // if (syscall == EXECVEAT) goto fail;
    /* 9 */ BPF_STMT( code: BPF_RET + BPF_K, SECCOMP_RET_ALLOW), // pass: return OK;
    /* 10 */ BPF_STMT( code: BPF_RET + BPF_K, k: SECCOMP_RET_ERRNO | (EACCES & SECCOMP_RET_DATA)), // fail: return EACCES;
};
```

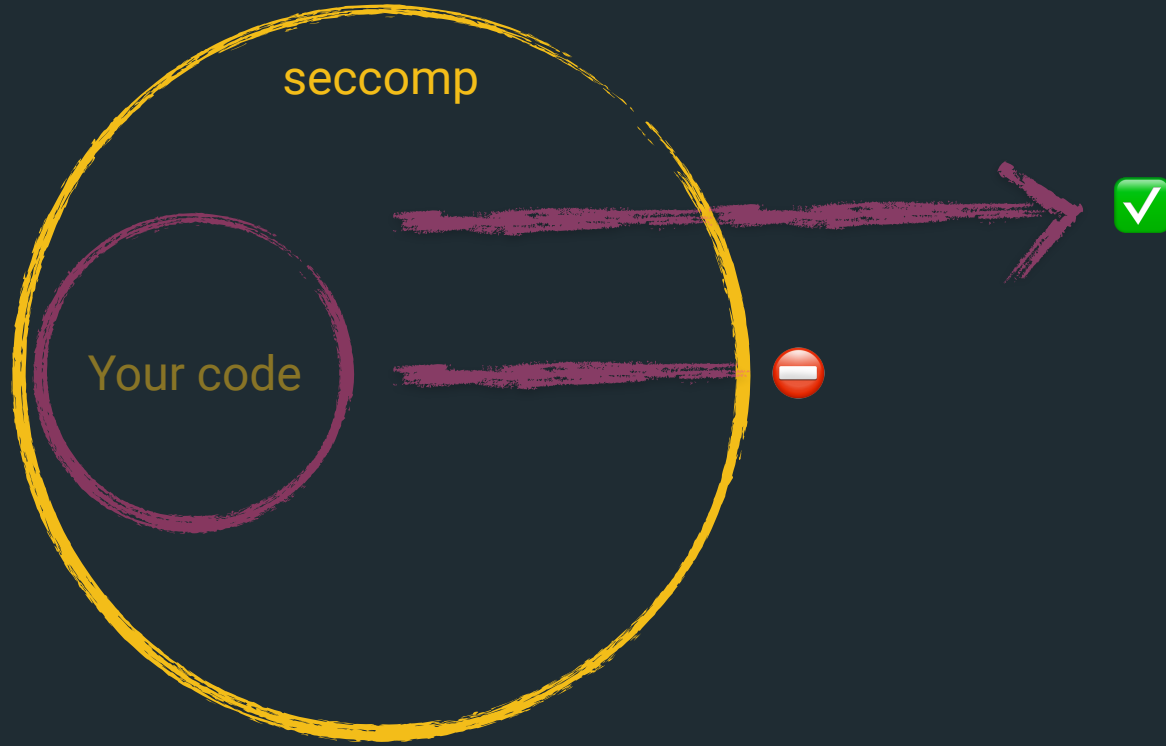
# Seccomp - prevent process forks

```
// seccomp takes a long, so we pass it one explicitly to keep the JNA simple
SockFProg prog = new SockFProg(insns);
prog.write();
long pointer = Pointer.nativeValue(prog.getPointer());

int method = 1;
// install filter, if this works, after this there is no going back!
// first try it with seccomp(SECCOMP_SET_MODE_FILTER), falling back to prctl()
if (linux_syscall(arch.seccomp, SECCOMP_SET_MODE_FILTER, SECCOMP_FILTER_FLAG_TSYNC, new NativeLong(pointer)) != 0) {
    method = 0;
    int errno1 = Native.getLastError();
    if (logger.isDebugEnabled()) {
        logger.debug("message: \"seccomp(SECCOMP_SET_MODE_FILTER): {}, falling back to prctl(PR_SET_SECCOMP)...\",
            JNACLibrary.sterror(errno1));
    }
}
if (linux_prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, pointer, arg4: 0, arg5: 0) != 0) {
    int errno2 = Native.getLastError();
    throw new UnsupportedOperationException("seccomp(SECCOMP_SET_MODE_FILTER): " + JNACLibrary.sterror(errno1) +
        ", prctl(PR_SET_SECCOMP): " + JNACLibrary.sterror(errno2));
}
}

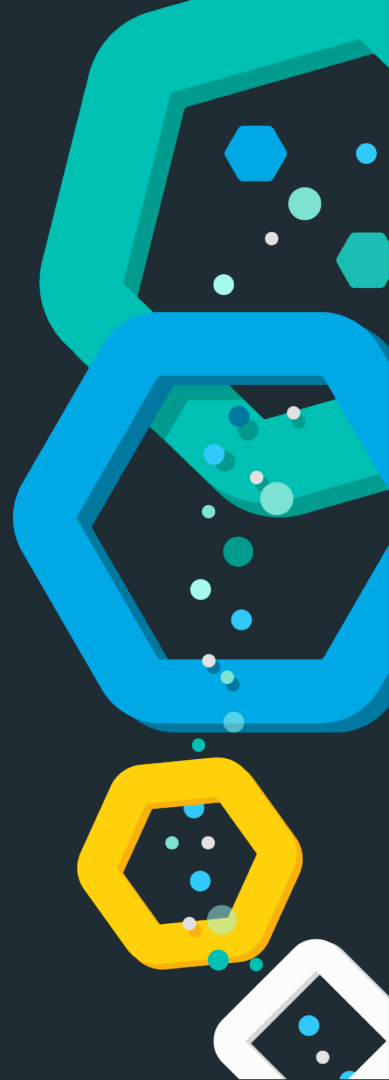
// now check that the filter was really installed, we should be in filter mode.
if (linux_prctl(PR_GET_SECCOMP, arg2: 0, arg3: 0, arg4: 0, arg5: 0) != 2) {
    throw new UnsupportedOperationException("seccomp filter installation did not really succeed. seccomp(PR_GET_SECCOMP): "
        + JNACLibrary.sterror(Native.getLastError()));
}
```

# seccomp sandbox





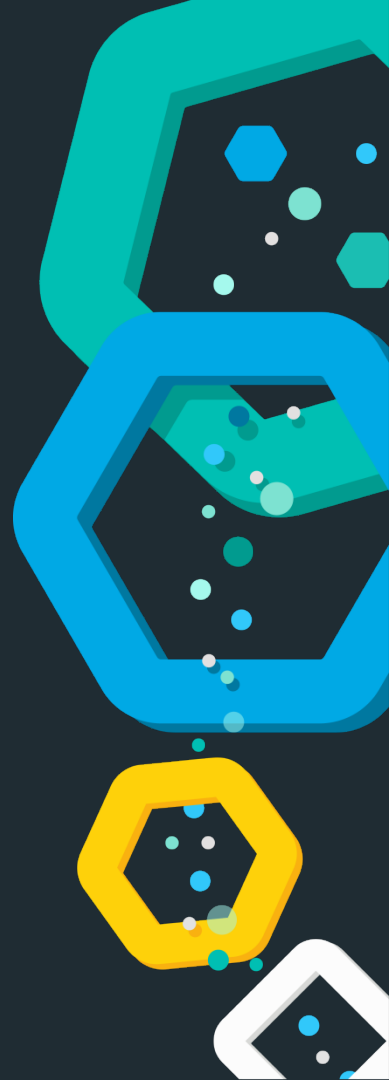
# DEMO





# Production mode vs Development mode

**Annoying you now instead of devastating you later**



# Bootstrap checks



# Is your dev setup equivalent to production?

- 🍄 Development environments are rarely setup like production ones
- 🍄 How to ensure certain preconditions in production but not for development?
- 🍄 What is a good indicator?

# Mode check

```
/**
 * Tests if the checks should be enforced.
 *
 * @param boundTransportAddress the node network bindings
 * @param discoveryType the discovery type
 * @return {@code true} if the checks should be enforced
 */
static boolean enforceLimits(final BoundTransportAddress boundTransportAddress, final String discoveryType) {
    final Predicate<TransportAddress> isLoopbackAddress = t -> t.address().getAddress().isLoopbackAddress();
    final boolean bound =
        !(Arrays.stream(boundTransportAddress.boundAddresses()).allMatch(isLoopbackAddress) &&
        isLoopbackAddress.test(boundTransportAddress.publishAddress()));
    return bound && !"single-node".equals(discoveryType);
}
```



# Bootstrap checks

```
// the list of checks to execute
static List<BootstrapCheck> checks() {
    final List<BootstrapCheck> checks = new ArrayList<>();
    checks.add(new HeapSizeCheck());
    final FileDescriptorCheck fileDescriptorCheck
        = Constants.MAC_OS_X ? new OsXFileDescriptorCheck()
    checks.add(fileDescriptorCheck);
    checks.add(new MlockallCheck());
    if (Constants.LINUX) {
        checks.add(new MaxNumberOfThreadsCheck());
    }
    if (Constants.LINUX || Constants.MAC_OS_X) {
        checks.add(new MaxSizeVirtualMemoryCheck());
    }
    if (Constants.LINUX || Constants.MAC_OS_X) {
        checks.add(new MaxFileSizeCheck());
    }
}
```

```
checks.add(new ClientJvmCheck());
checks.add(new UseSerialGCCheck());
checks.add(new SystemCallFilterCheck());
checks.add(new OnErrorCheck());
checks.add(new OnOutOfMemoryErrorCheck());
checks.add(new EarlyAccessCheck());
checks.add(new G1GCCheck());
checks.add(new AllPermissionCheck());
return Collections.unmodifiableList(checks);
}
```

# Bootstrap checks

```
static class FileDescriptorCheck implements BootstrapCheck {  
  
    private final int limit;  
  
    FileDescriptorCheck() { this( limit: 65535); }  
  
    protected FileDescriptorCheck(final int limit) {  
        if (limit <= 0) {  
            throw new IllegalArgumentException("limit must be positive but was [" + limit + "]);  
        }  
        this.limit = limit;  
    }  
  
    public final BootstrapCheckResult check(BootstrapContext context) {  
        final long maxFileDescriptorCount = getMaxFileDescriptorCount();  
        if (maxFileDescriptorCount != -1 && maxFileDescriptorCount < limit) {  
            final String message = String.format(  
                Locale.ROOT,  
                format: "max file descriptors [%d] for elasticsearch process is too low, increase to at least [%d]",  
                getMaxFileDescriptorCount(),  
                limit);  
            return BootstrapCheckResult.failure(message);  
        } else {  
            return BootstrapCheckResult.success();  
        }  
    }  
  
    // visible for testing  
    long getMaxFileDescriptorCount() { return ProcessProbe.getInstance().getMaxFileDescriptorCount(); }  
}
```

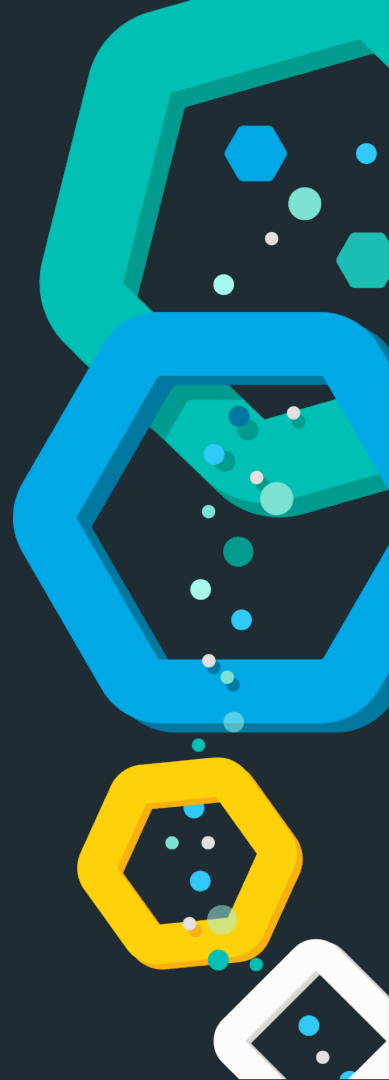
# Bootstrap checks

```
/**
 * Bootstrap check for versions of HotSpot that are known to have issues that can lead to index corruption when G1GC is enabled.
 */
static class G1GCCheck implements BootstrapCheck {

    @Override
    public BootstrapCheckResult check(BootstrapContext context) {
        if ("Oracle Corporation".equals(jvmVendor()) && isJava8() && isG1GCEnabled()) {
            final String jvmVersion = jvmVersion();
            // HotSpot versions on Java 8 match this regular expression; note that this changes with Java 9 after JEP-223
            final Pattern pattern = Pattern.compile("(\\d+)\\.\\.((\\d+)-b\\d+)");
            final Matcher matcher = pattern.matcher(jvmVersion);
            final boolean matches = matcher.matches();
            assert matches : jvmVersion;
            final int major = Integer.parseInt(matcher.group(1));
            final int update = Integer.parseInt(matcher.group(2));
            // HotSpot versions for Java 8 have major version 25, the bad versions are all versions prior to update 40
            if (major == 25 && update < 40) {
                final String message = String.format(
                    Locale.ROOT,
                    format: "JVM version [%s] can cause data corruption when used with G1GC; upgrade to at least Java 8u40", jvmVersion);
                return BootstrapCheckResult.failure(message);
            }
        }
        return BootstrapCheckResult.success();
    }
}
```

# Plugins

... remaining secure





# Bootstrap checks



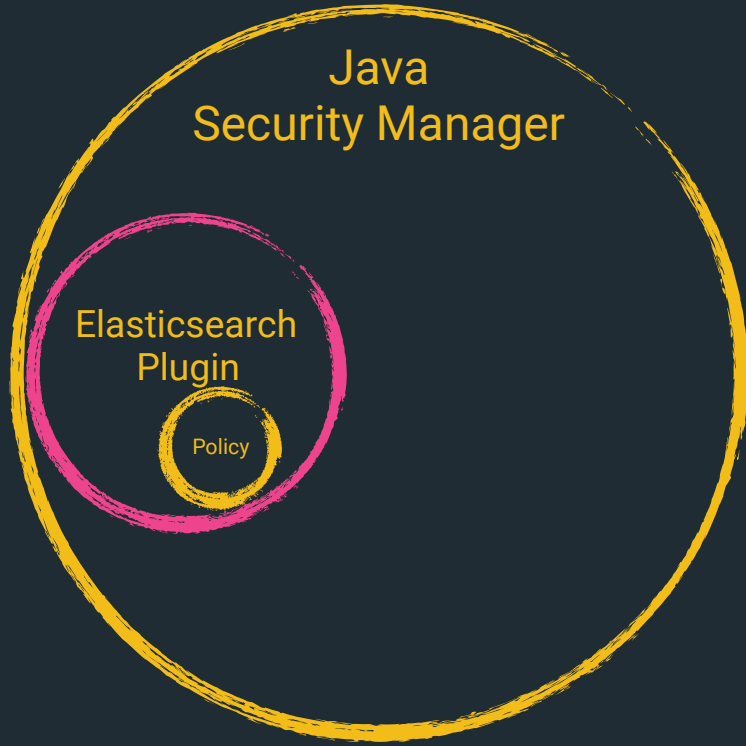
# Plugins in 60 seconds

- 🌿 plugins are just zip files
- 🌿 each plugin can have its own jars/dependencies
- 🌿 each plugin is loaded with its own classloader
- 🌿 each plugin can have its own security permissions
- 🌿 ES core loads a bunch of code as modules (plugins that ship with Elasticsearch)

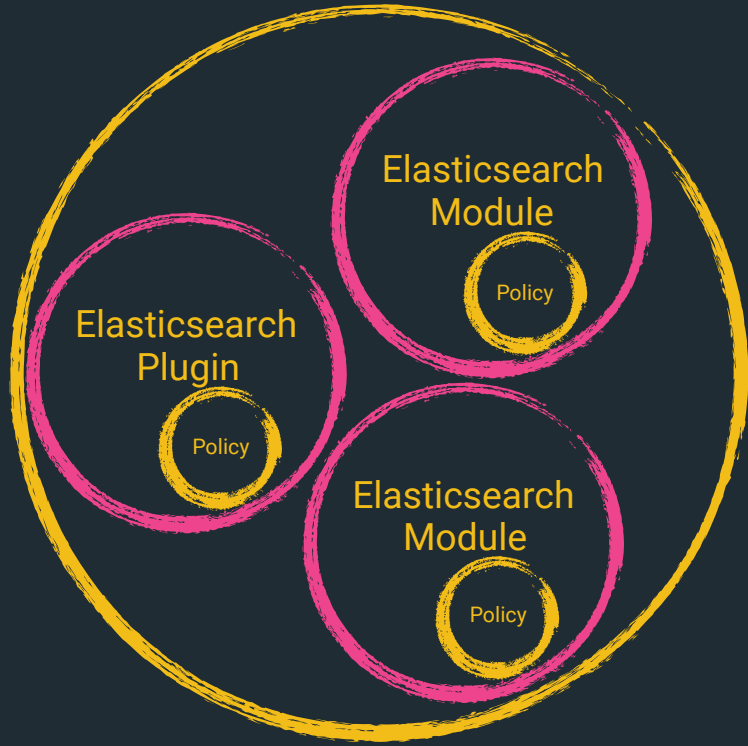
# Plugins & modules



# Plugins & modules



# Plugins & modules



# Sample permissions

```
grant {  
    // needed to do crazy reflection  
    permission java.lang.RuntimePermission "accessDeclaredMembers";  
};
```

# Sample permissions

```
grant {  
  // needed to generate runtime classes  
  permission java.lang.RuntimePermission "createClassLoader";  
  
  // expression runtime  
  permission org.elasticsearch.script.ClassPermission "java.lang.String";  
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.expressions.Expression";  
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.search.DoubleValues";  
  // available functions  
  permission org.elasticsearch.script.ClassPermission "java.lang.Math";  
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.util.MathUtil";  
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.util.SloppyMath";  
};
```

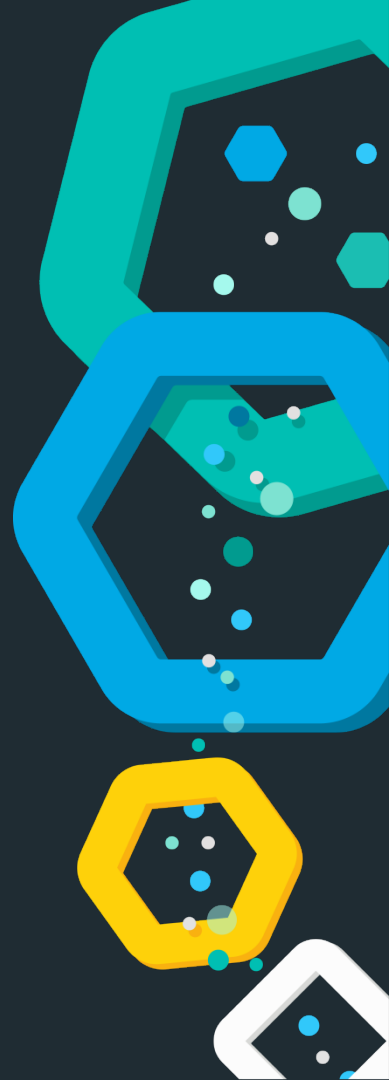
# Sample permissions

```
grant codeBase "${codebase.netty-common}" {  
    // for reading the system-wide configuration for the backlog of established sockets  
    permission java.io.FilePermission "/proc/sys/net/core/somaxconn", "read";  
  
    // netty makes and accepts socket connections  
    permission java.net.SocketPermission "*", "accept,connect";  
};  
  
grant codeBase "${codebase.netty-transport}" {  
    // Netty NioEventLoop wants to change this, because of https://bugs.openjdk.java.net/browse/JDK-6427854  
    // the bug says it only happened rarely, and that its fixed, but apparently it still happens rarely!  
    permission java.util.PropertyPermission "sun.nio.ch.bugLevel", "write";  
};
```



# Introducing Painless

A scripting language for Elasticsearch



# Scripting: Why and how?

- 🌀 Expression evaluation without needing to write java extensions for Elasticsearch
- 🌀 Node ingest script processor
- 🌀 Search queries (dynamic requests & fields)
- 🌀 Aggregations (dynamic buckets)
- 🌀 Templating (Mustache)

# Scripting in Elasticsearch

 MVEL

 Groovy

 Expressions

 Painless

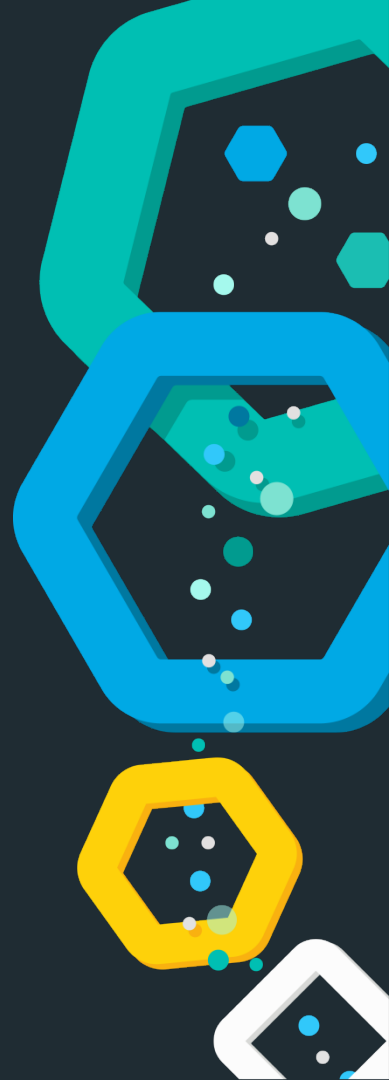


# Painless - a secure scripting language

- 🍄 Hard to take an existing programming language and make it secure, but remain fast
- 🍄 Sandboxing
- 🍄 Whitelisting over blacklisting, per method
- 🍄 Opt-in to regular expressions
- 🍄 Prevent endless loops
- 🍄 Detect self references to prevent stack overflows



# DEMO



# Summary

**Security is hard - let's go shopping!**



# Summary

- 🌸 Not using the Security Manager - what's your excuse?
- 🌸 Scripting is important, is your implementation secure?
- 🌸 Use operating system features!
- 🌸 If you allow for plugins, remain secure!
- 🌸 If you remove features, have alternatives!

# Summary

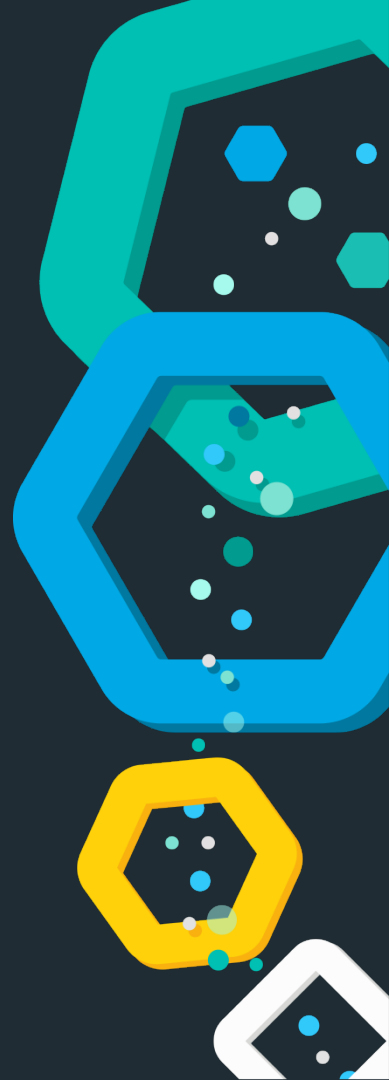
- 🍄 Development has big impact on security
- 🍄 Operations is happy to help what is there out of the box
- 🍄 Developers know their application best!
- 🍄 Don't reinvent, check out existing features!
- 🍄 Developers are responsible for writing secure code! Before something happens!



# Thanks for listening!

## Questions?

Alexander Reelsen  
@spinscale  
alex@elastic.co



# Resources

spinscale / talk-elasticsearch-security-manager-and-seccomp

Unwatch 1 Star 0 Fork 0


Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Corresponding demos to my talk about Elasticsearch, its use of the Java Security Manager & seccomp [Edit](#)

[Manage topics](#)

2 commits 1 branch 0 packages 0 releases 1 contributor Apache-2.0

Branch: master New pull request Create new file Upload files Find File Clone or download

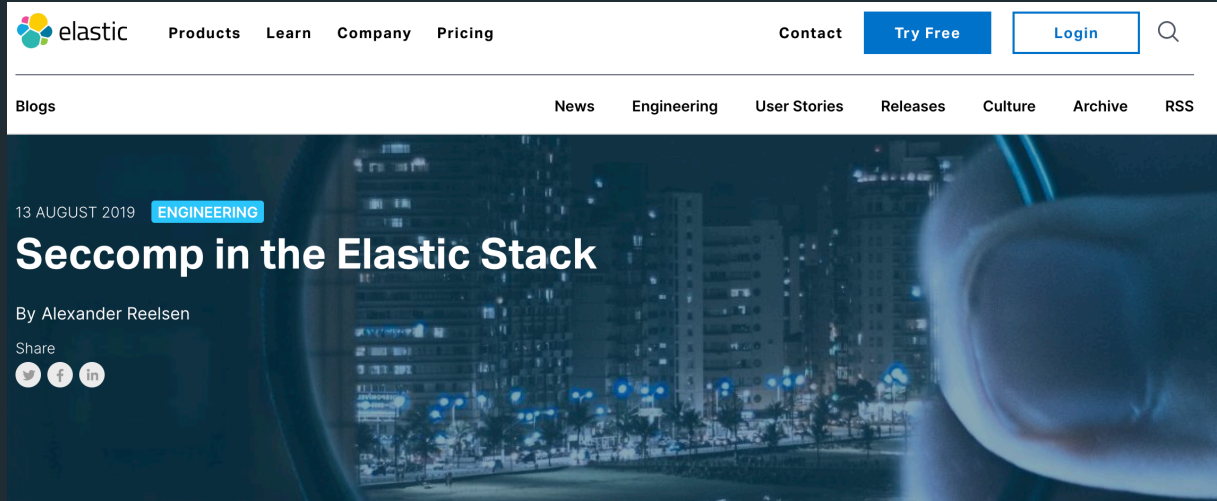
 spinscale Add vagrant command to docs Latest commit 183f6d6 3 days ago

<a href="#">gradle/wrapper</a>	Initial commit	3 days ago
<a href="#">seccomp-example-vm</a>	Initial commit	3 days ago
<a href="#">src/main/java/de/spinscale/security/samples</a>	Initial commit	3 days ago
<a href="#">.gitignore</a>	Initial commit	3 days ago
<a href="#">LICENSE.txt</a>	Initial commit	3 days ago
<a href="#">README.md</a>	Add vagrant command to docs	3 days ago
<a href="#">build.gradle</a>	Initial commit	3 days ago
<a href="#">gradlew</a>	Initial commit	3 days ago
<a href="#">gradlew.bat</a>	Initial commit	3 days ago
<a href="#">painless.snippets</a>	Initial commit	3 days ago
<a href="#">settings.gradle</a>	Initial commit	3 days ago

README.md

## Elasticsearch, the Java Security Manager and seccomp

# Resources



The screenshot shows the Elastic website's navigation bar with links for Products, Learn, Company, Pricing, Contact, Try Free, and Login. Below the navigation bar is a secondary menu with links for Blogs, News, Engineering, User Stories, Releases, Culture, Archive, and RSS. The main content area features a featured article titled "Seccomp in the Elastic Stack" by Alexander Reelsen, dated 13 August 2019, with an "ENGINEERING" tag. The article is set against a background image of a city at night with a hand holding a pen in the foreground.

13 AUGUST 2019

ENGINEERING

## Seccomp in the Elastic Stack

By Alexander Reelsen

Share



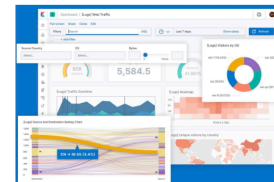
After giving a presentation about what is done in [Elasticsearch to improve out-of-the-box security, safety and usability](#) and engaging in a couple of follow-up discussions at different events, I decided to dig a little bit deeper into the topic of Linux's seccomp.

### What is seccomp?

The idea of seccomp is to prevent the execution of certain system calls by a given application.

Why is this kind of security feature needed? Imagine someone finds a way to execute code within your application, which also implies they have the same user rights through which the application was started. Common use cases for this are web applications, where malicious attackers find a way to execute arbitrary commands — often due to invalid validation of input arguments. These attacks are categorized as remote

### Creating Kibana Dashboards



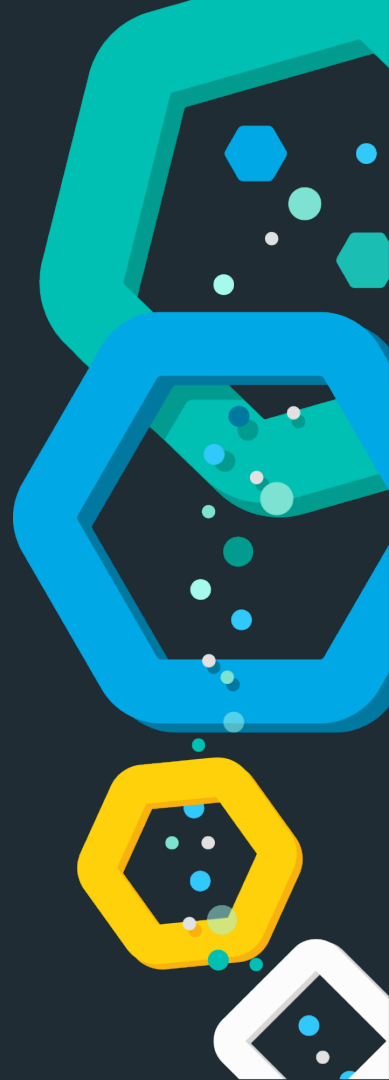
Watch our demo on customizing and optimizing your visualizations.

# Resources

- 🌸 <https://github.com/elastic/elasticsearch/>
- 🌸 [https://www.elastic.co/blog/bootstrap\\_checks\\_annoying\\_instead\\_of\\_devastating](https://www.elastic.co/blog/bootstrap_checks_annoying_instead_of_devastating)
- 🌸 <https://www.elastic.co/blog/scripting>
- 🌸 <https://www.elastic.co/blog/scripting-security>
- 🌸 <https://docs.oracle.com/javase/9/security/toc.htm>
- 🌸 <https://docs.oracle.com/javase/9/security/permissions-java-development-kit.htm>
- 🌸 <https://www.elastic.co/blog/seccomp-in-the-elastic-stack>
- 🌸 <https://github.com/spinscale/talk-elasticsearch-security-manager-and-seccomp>

# Bonus


**register all your settings**



# Mark sensitive settings

```
/**
 * Username for basic auth.
 */
public static final Setting.AffixSetting<String> AUTH_USERNAME_SETTING =
    Setting.affixKeySetting( prefix: "xpack.monitoring.exporters.", suffix: "auth.username",
        (key) -> Setting.simpleString(key, Property.Dynamic, Property.NodeScope, Property.Filtered));
/**
 * Password for basic auth.
 */
public static final Setting.AffixSetting<String> AUTH_PASSWORD_SETTING =
    Setting.affixKeySetting( prefix: "xpack.monitoring.exporters.", suffix: "auth.password",
        (key) -> Setting.simpleString(key, Property.Dynamic, Property.NodeScope, Property.Filtered));

private static final Setting.AffixSetting<SecureString> SETTING_URL_SECURE =
    Setting.affixKeySetting( prefix: "xpack.notification.slack.account.", suffix: "secure_url",
        (key) -> SecureSetting.secureString(key, fallback: null));
```



# Register all your settings

```
stacks/7.1.1/elasticsearch-7.1.1 bin/elasticsearch -Ecluster.namr=my-cluster
```



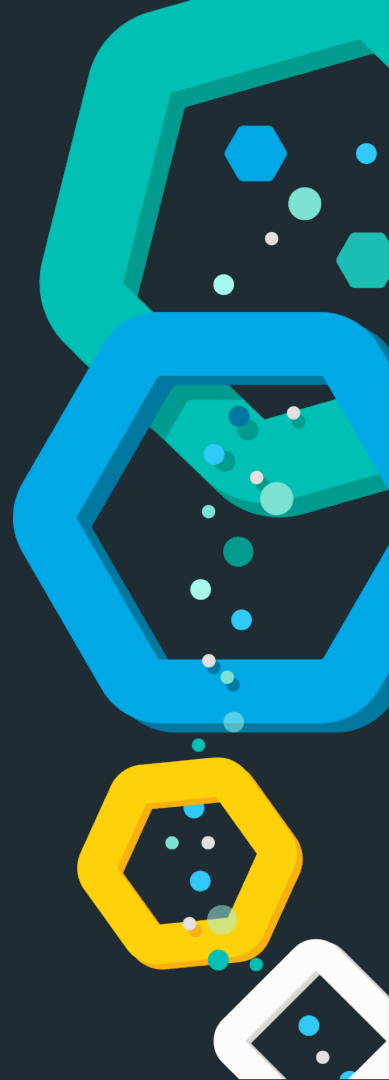
```
[2019-06-21T10:42:56,943][WARN ][o.e.b.ElasticsearchUncaughtExceptionHandler] [rhincodon] uncaught exception in thread [main]  
org.elasticsearch.bootstrap.StartupException: java.lang.IllegalArgumentException: unknown setting [cluster.namr] did you mean [cluster.name]?  
    at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:163) ~[elasticsearch-7.1.1.jar:7.1.1]  
    at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:150) ~[elasticsearch-7.1.1.jar:7.1.1]  
    at org.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:86) ~[elasticsearch-7.1.1.jar:7.1.1]
```



```
unknown setting [cluster.namr] did you mean [cluster.name]?
```

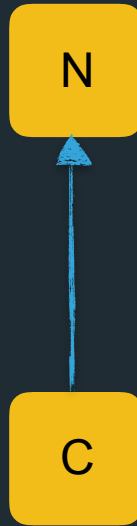
# Bonus

deep pagination vs search\_after





# Pagination: Request



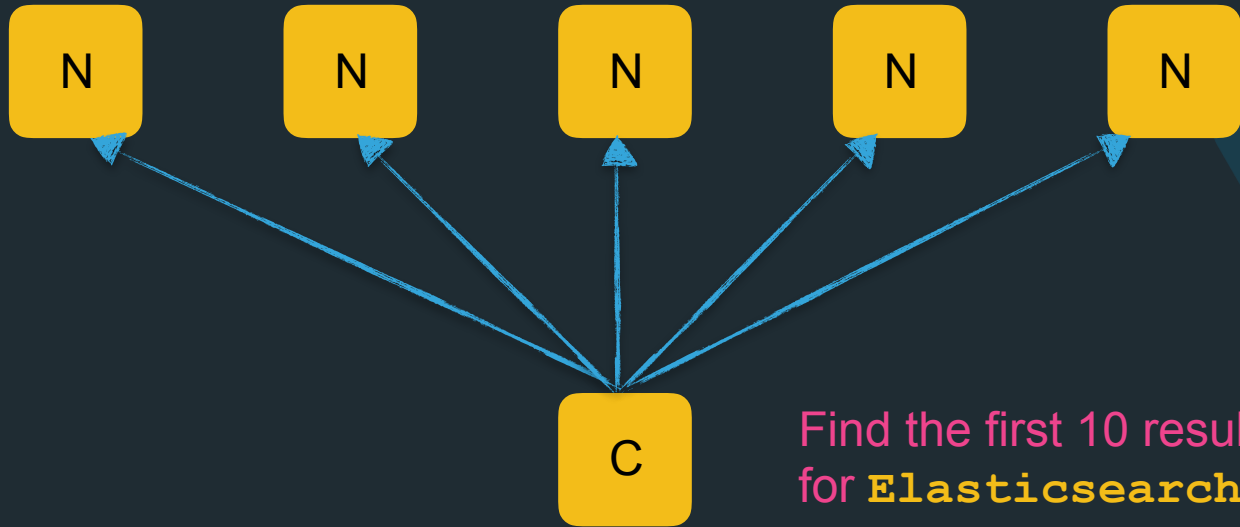
Find the first 10 results  
for **Elasticsearch**

# Pagination: Request

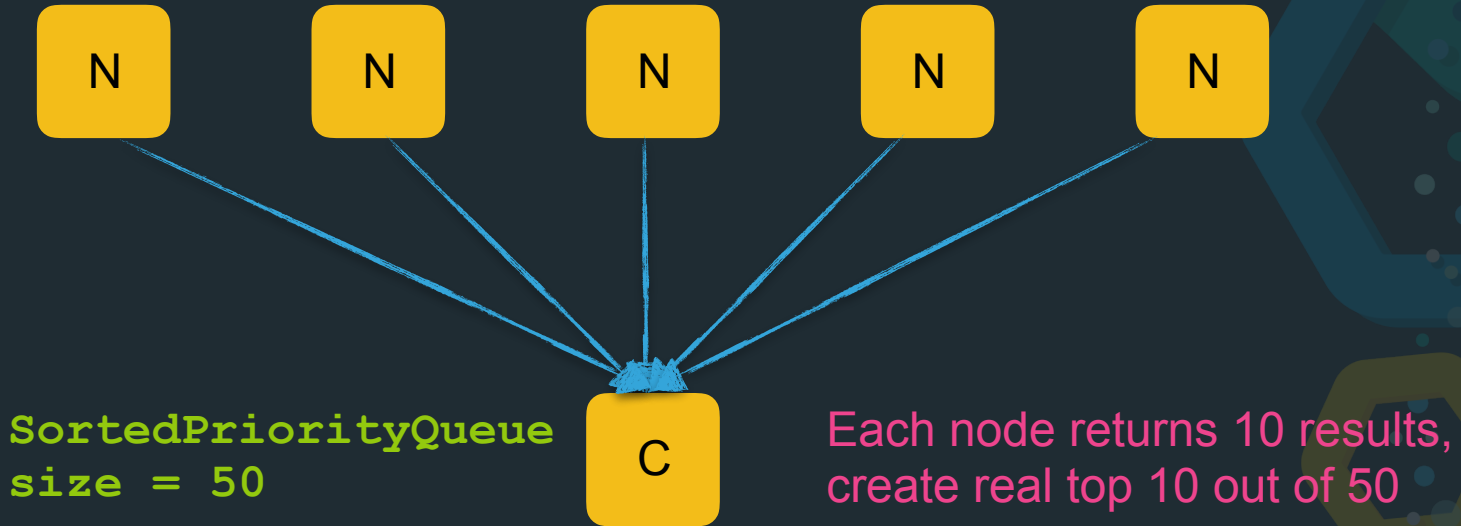


Find the first 10 results  
for **Elasticsearch**

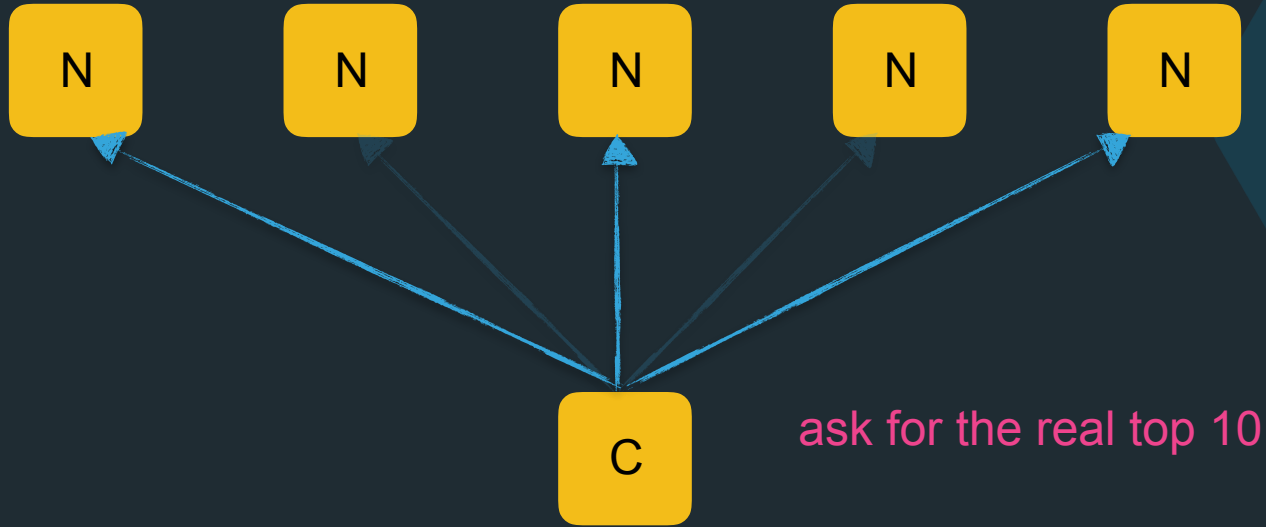
# Pagination: Request



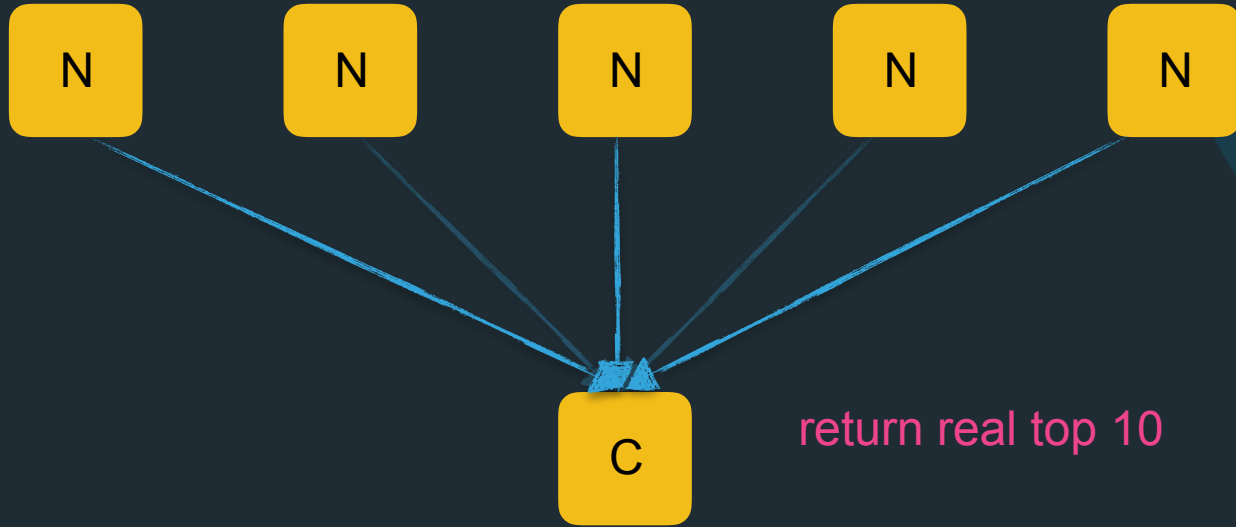
# Pagination: Query Phase



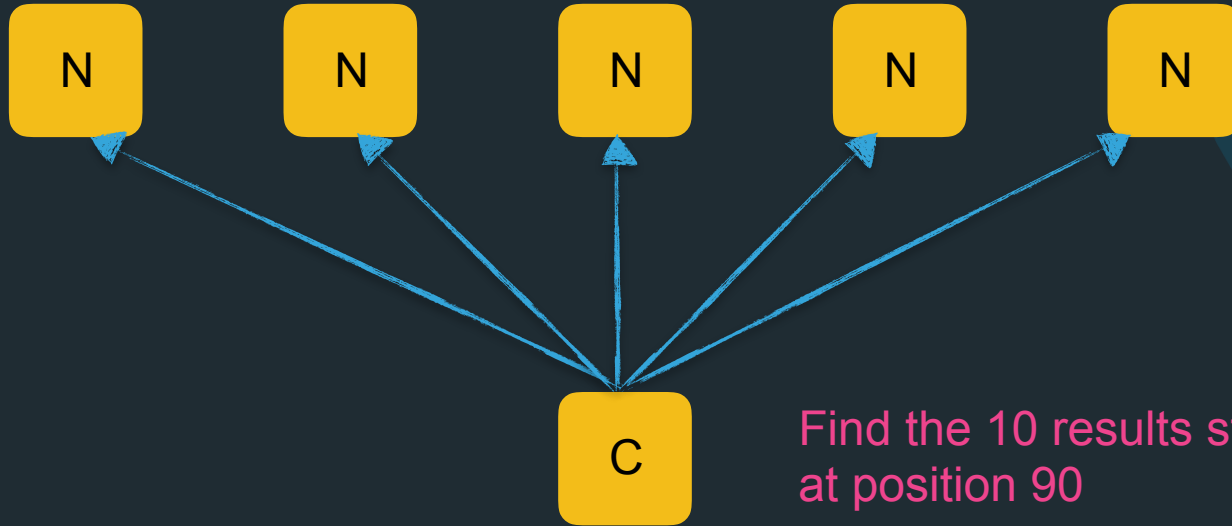
# Pagination: Fetch phase



# Pagination: Query Phase

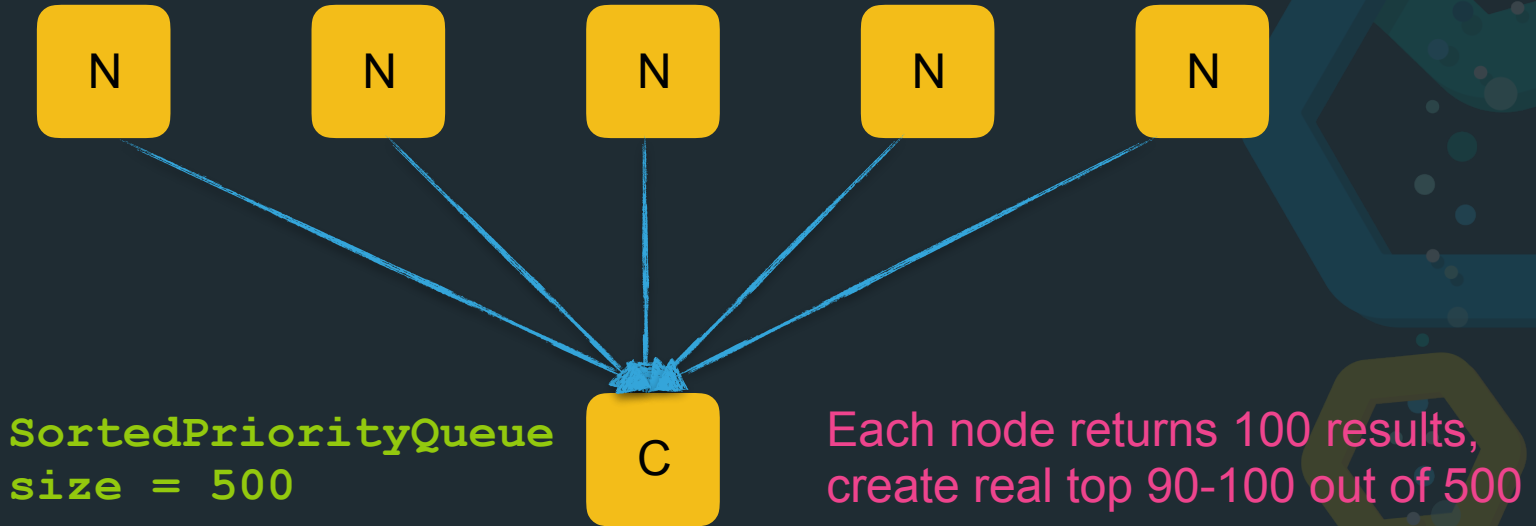


# Pagination: Query



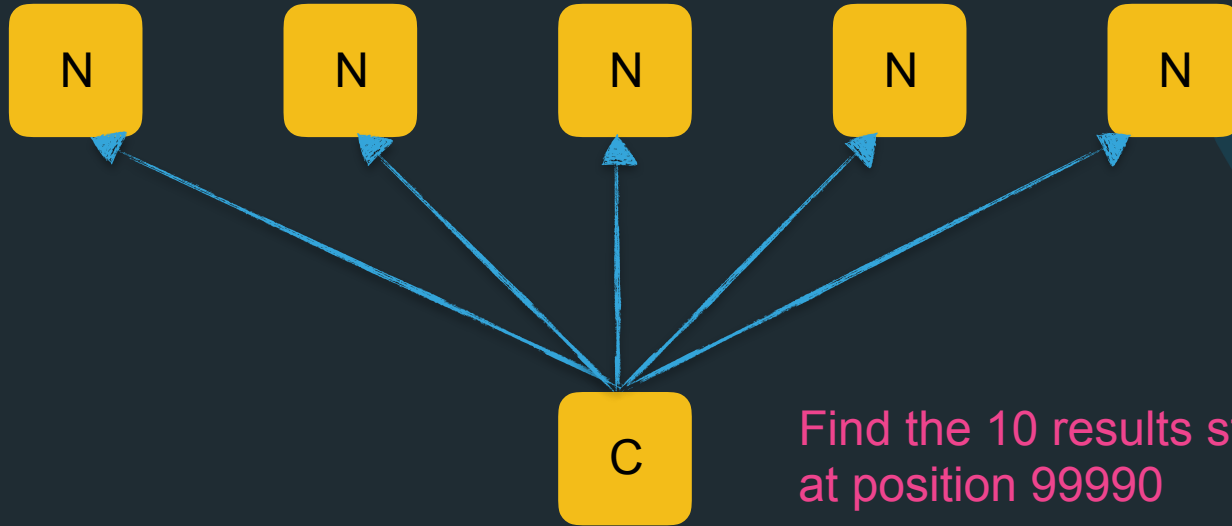
Find the 10 results starting  
at position 90

# Pagination: Query Phase



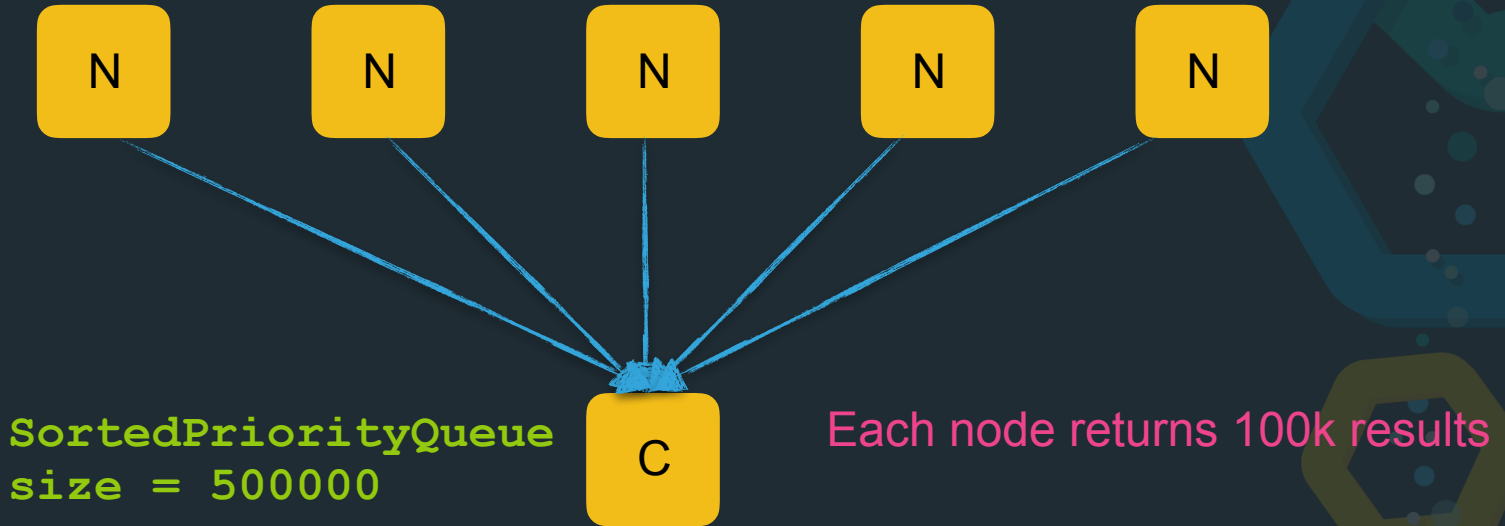


# Pagination: Query

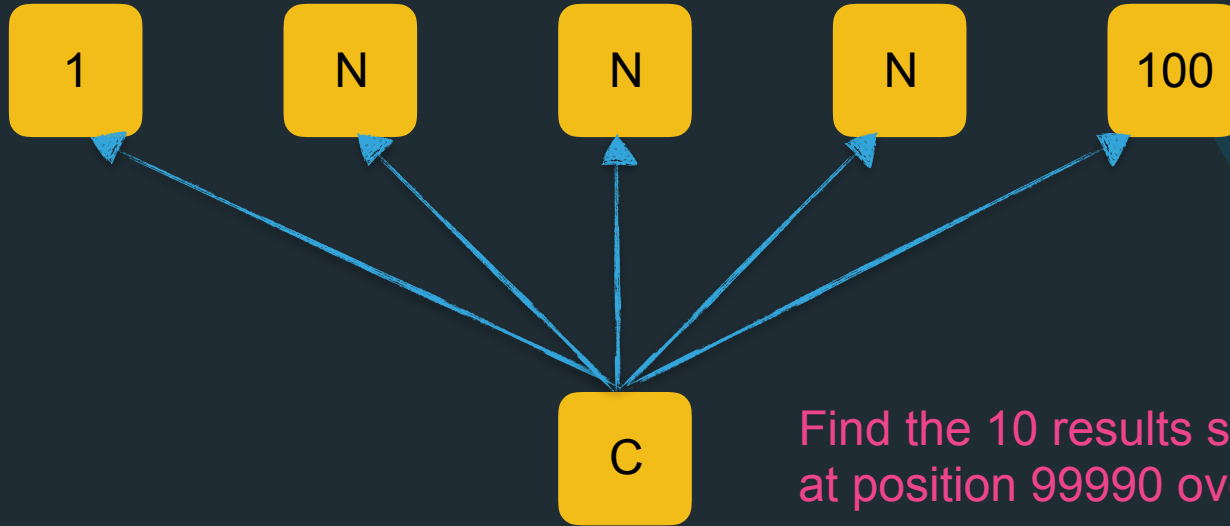


Find the 10 results starting  
at position 99990

# Pagination: Query Phase

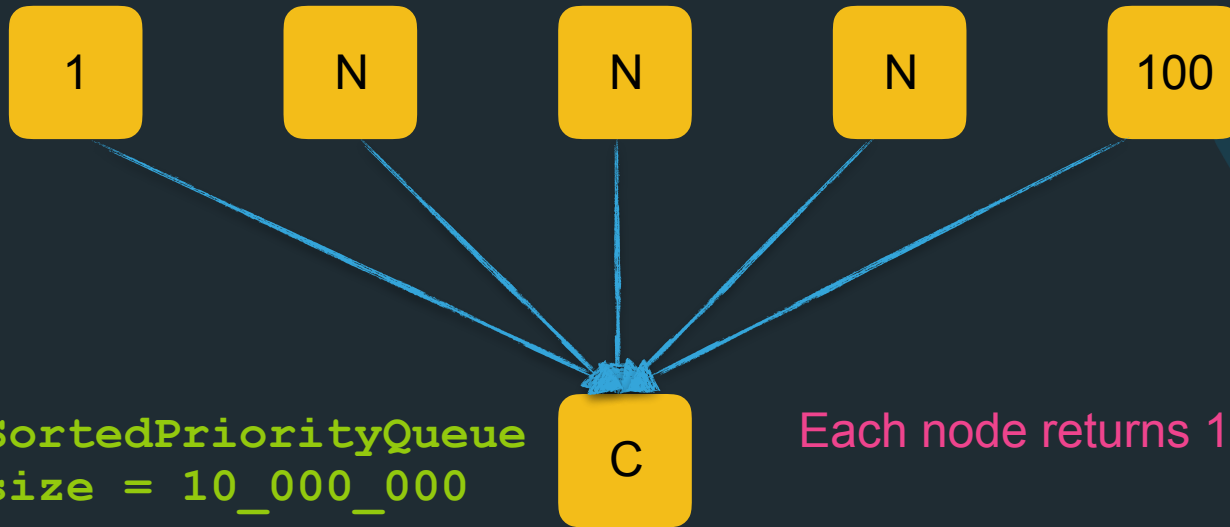


# Pagination: Query



Find the 10 results starting  
at position 99990 over 100 nodes

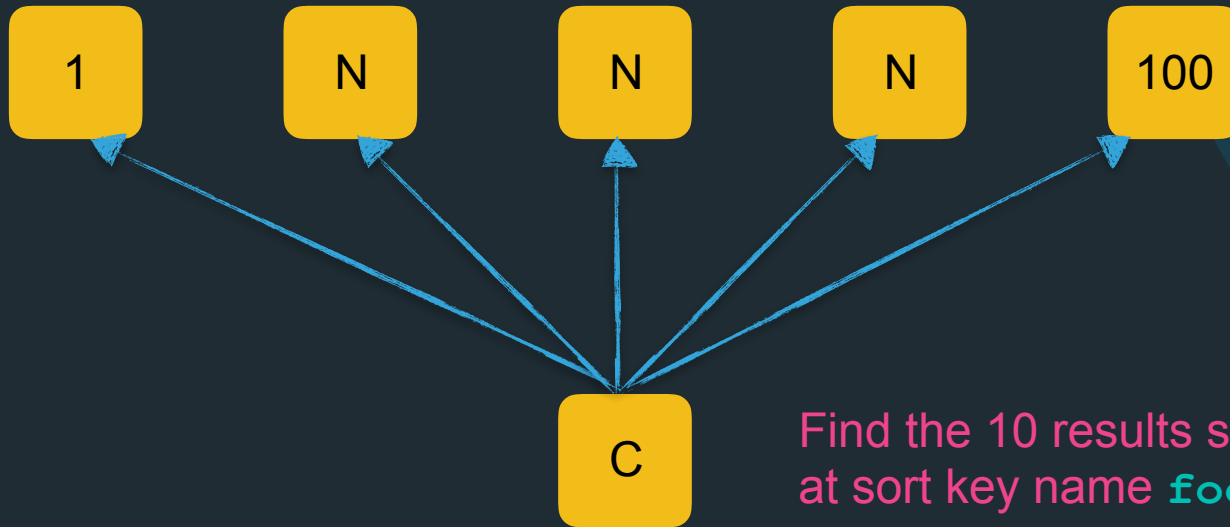
# Pagination: Query



# Solution: search\_after

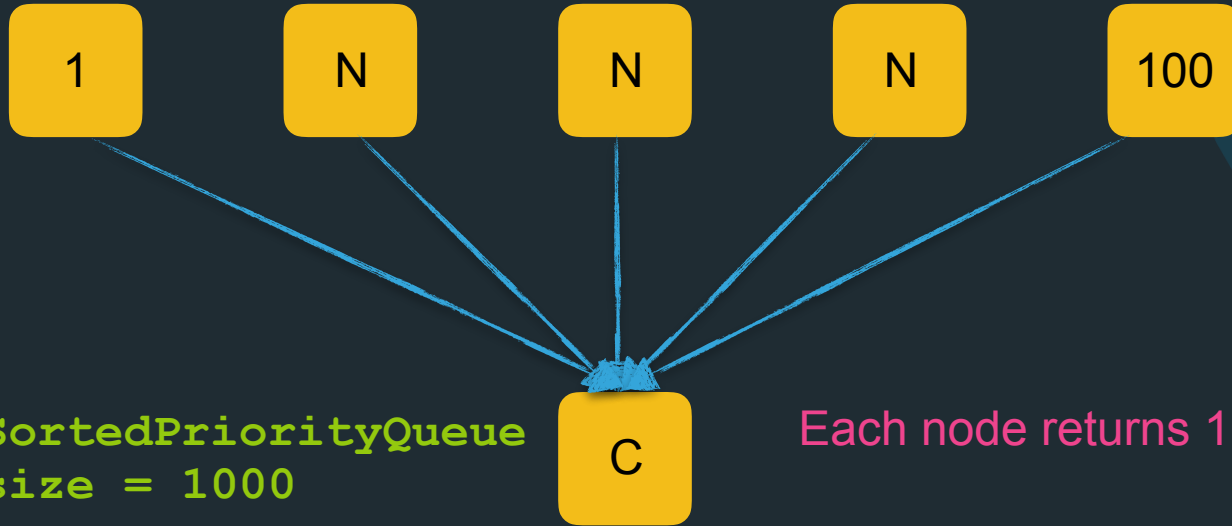
- 🌿 Do not use numerical positions
- 🌿 Use keys where you stopped in the inverted index
- 🌿 Let the client tell you what the last key was
- 🌿 Just specify the last sort value from the last document returned as a starting point

# Pagination: search\_after



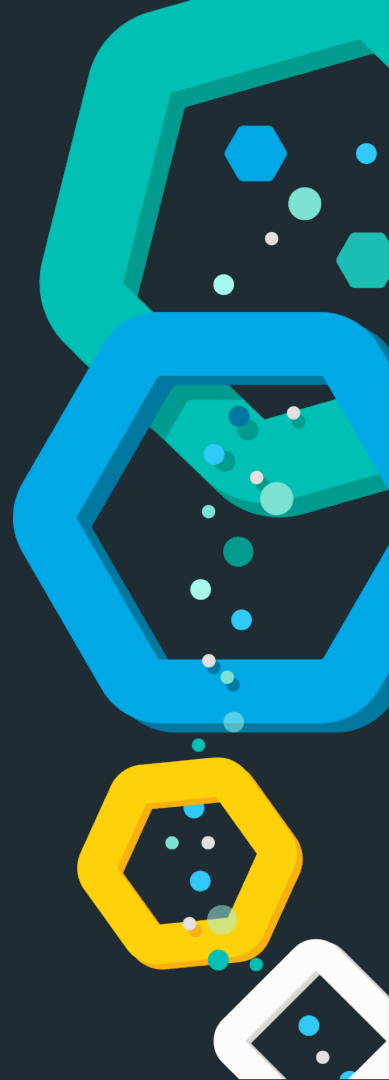
Find the 10 results starting  
at sort key name `foo` over  
100 nodes

# Pagination: search\_after



# Bonus

replacing delete by query





# delete\_by\_query removal/replace

- 🌀 **delete\_by\_query** API was not safe
- 🌀 API endpoint was removed
- 🌀 extensive documentation was added what to do instead
- 🌀 infrastructure for long running background tasks was added
- 🌀 **delete\_by\_query** was reintroduced using above infra and doing the exact same thing as in the documentation
- 🌀 data > convenience!

# Thanks for listening!

## Questions?

Alexander Reelsen  
@spinscale  
alex@elastic.co

