JavaLand

March 15 - 17, 2022

in Phantasialand
near Cologne

Jakarta EE Security

Sailing Safe in Troubled Waters

Werner Keil, Ivar Grimstad

www.javaland.eu

Agenda

1. Motivation
2. Use Cases
3. Specifications
4. Way Forward
5. Demos
6. Q&A

Who are we?

[www.linkedin.com/in/catmedia]

Werner Keil

- Consultant – Coach
- Creative Cosmopolitan
- Open Source Evangelist
- Software Architect
- Author, Speaker
- Maintenance Lead – JSR 354, 385
- Jakarta EE Spec Committee Member

ECLIPSE FOUNDATION

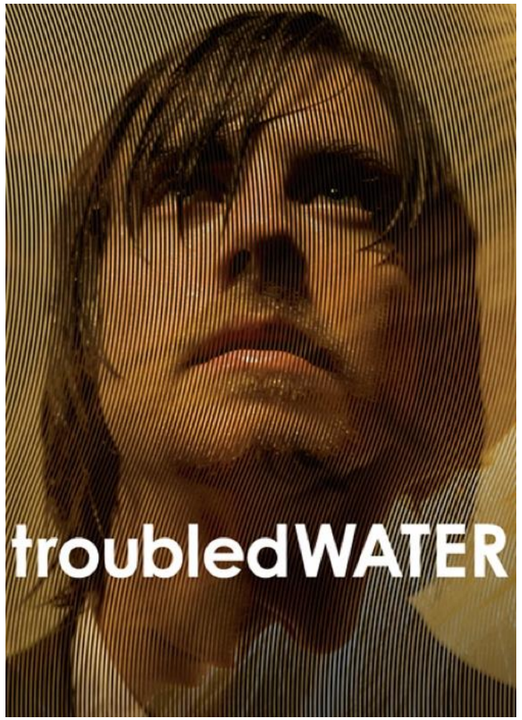
Ivar Grimstad

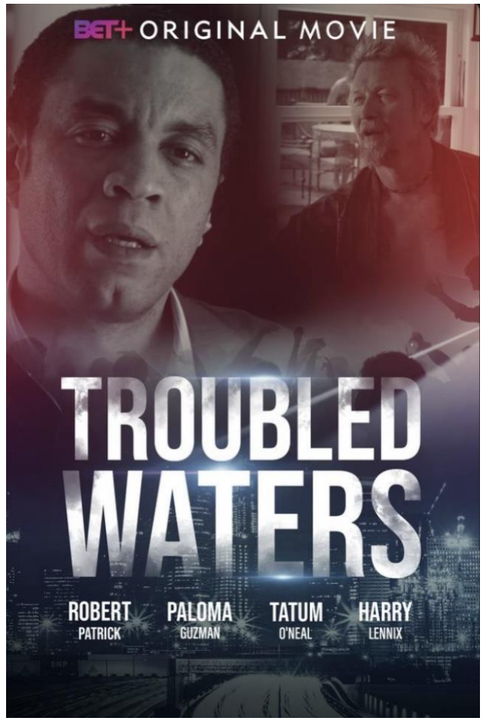
- Jakarta EE Developer Advocate
- Java Champion
- JUG Leader
- JCP Executive Committee Member

JAKARTA EE

Motivation

Troubled Water(s)



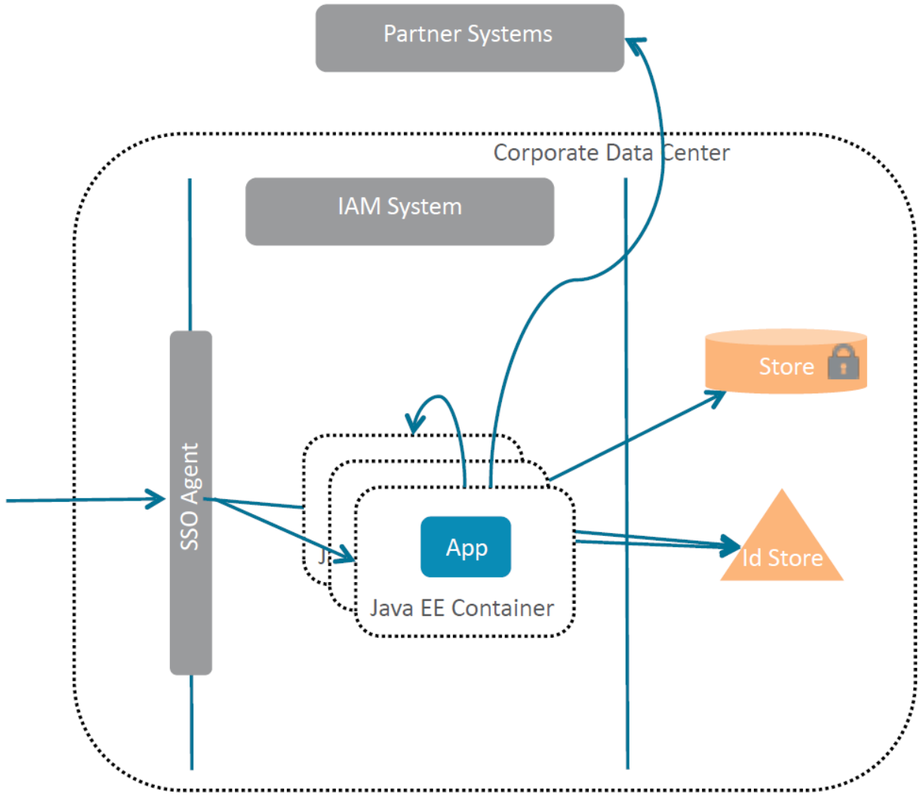


Motivation

- Where Enterprise apps run is changing
  - In corporate data centers
  - In the cloud from one of several vendors
- The shape of the Enterprise app is changing
  - A monolith or a collection of microservices
  - In a container or “serverless”
- These factors
  - Drive complexity in how apps are built, deployed, managed, operated
  - Drive complexity in how apps need to work in their target environment
- Can we still stay secure after these changes?

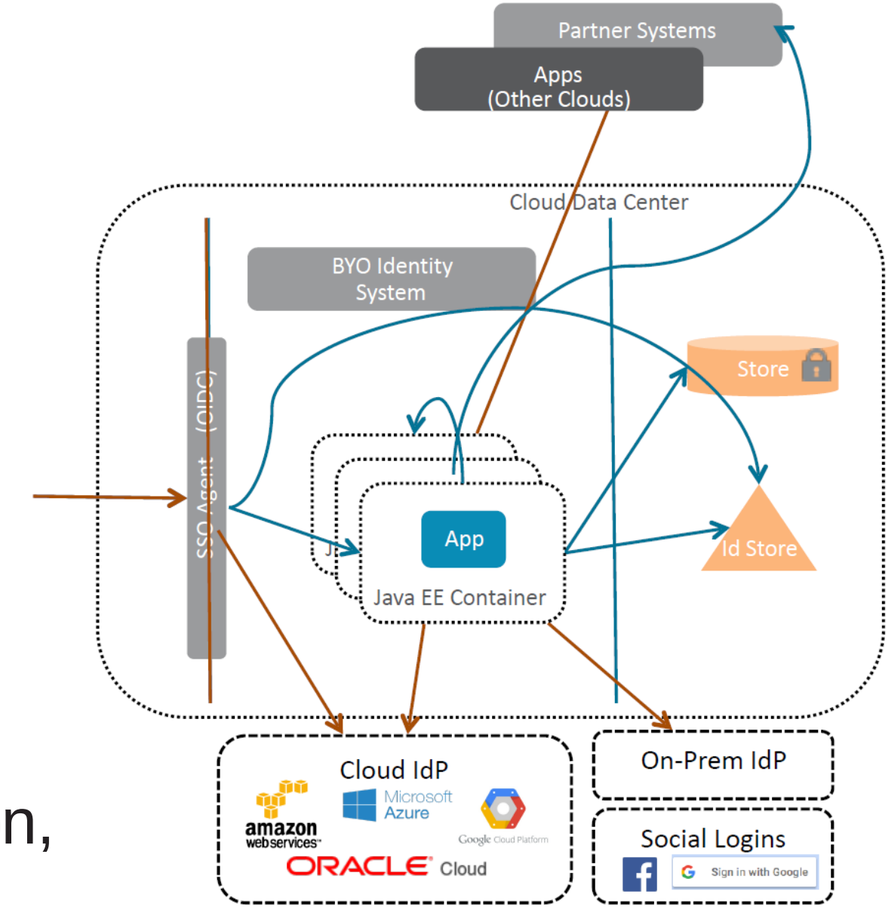
Deployed On premise

- Deployed within the corporate network
- Authenticates to on premise identity systems
- May use on premise Single Sign-on to secure web resources
- Authorization : managed by application, mapped to on premise identity
- Identity propagation to external entities relies on SAML, Basic Auth
- Secrets in local stores with several layers of control



Deployed in the Cloud

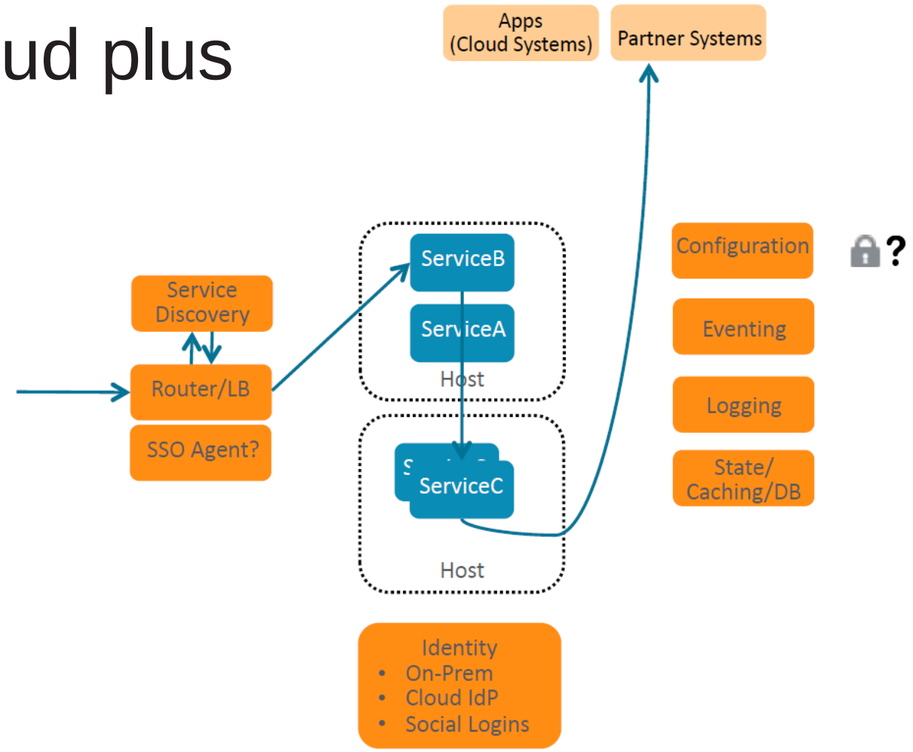
- Cloud Vendor for controls on network
- Social logins, external Identity Systems
- SSO using a Cloud Identity provider
- REST needs OAuth
- Identity Propagation - SAML,
  Basic Auth plus OAuth, OIDC and JWT
- More interactions – cloud, on premise
- Authorization - from one of several
  identity providers
- Secrets need defense in depth – encryption,
  securing the encryption key?



Microservices in the Cloud

- All issues of Jakarta EE App in the Cloud plus
- App Boundary is changing
  - Distributed processes, scale independently
  - Identity on every hop?
  - Each microservice deals with identity?
  - Each microservice authorizes access?
  - Each microservice manages secrets?
  - What about statelessness, configuration ?
  - What about the network boundary?

    Which microservices are public?



Use Case

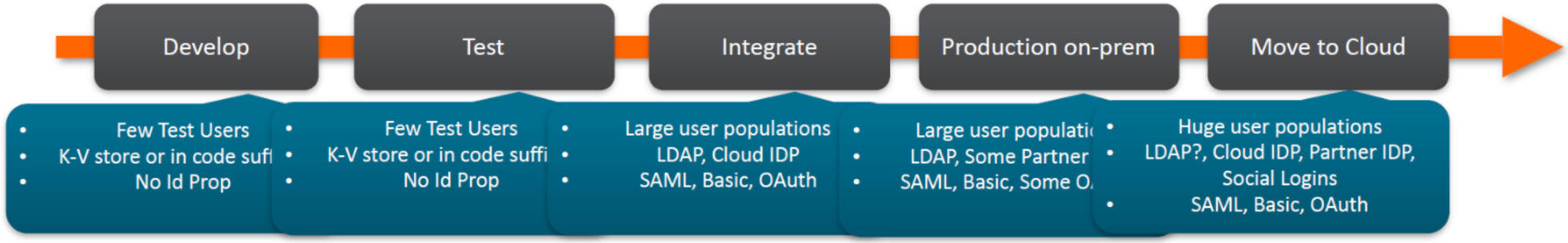
Authentication

- Application may manage its users or use externally managed users
- Application must authenticate users against one of several identity stores
- Application must support one of these authentication methods
  - Basic Auth, OpenID Connect
- Application is able to handle Authentication events (login, logout)
- Developer is able to use a portable Authentication API regardless of the identity store

Use Case

Identity Store

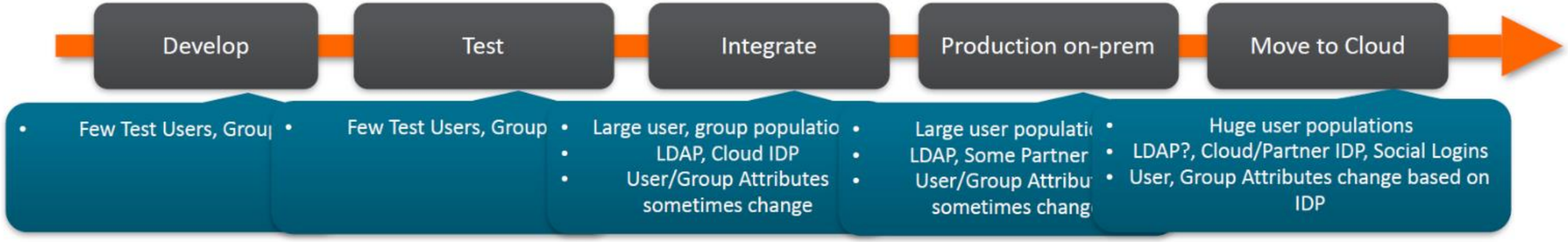
- Application may manage its users or use externally managed users
- Application must be able access the identity store
- Application can be bound to one or more identity stores at deployment
- Identity Store bound to the Application can be reconfigured



Use Case

Identity Representation

- Application must be able to determine identity of the caller
- Application is able to determine user's groups.
- Application knows caller identity consistently, as identity stores change



Use Case

Security Context

- Application is able to determine user attributes consistently
  - Authenticated user
  - Groups, Roles
  - Identity Provider that issued claims used in creating the Subject
  - Local or remote user? Virtual User?
- Application needs a consistent API to access security context

Jakarta EE Security Specs

- Authentication
- Authorization
- Security

Common Principles

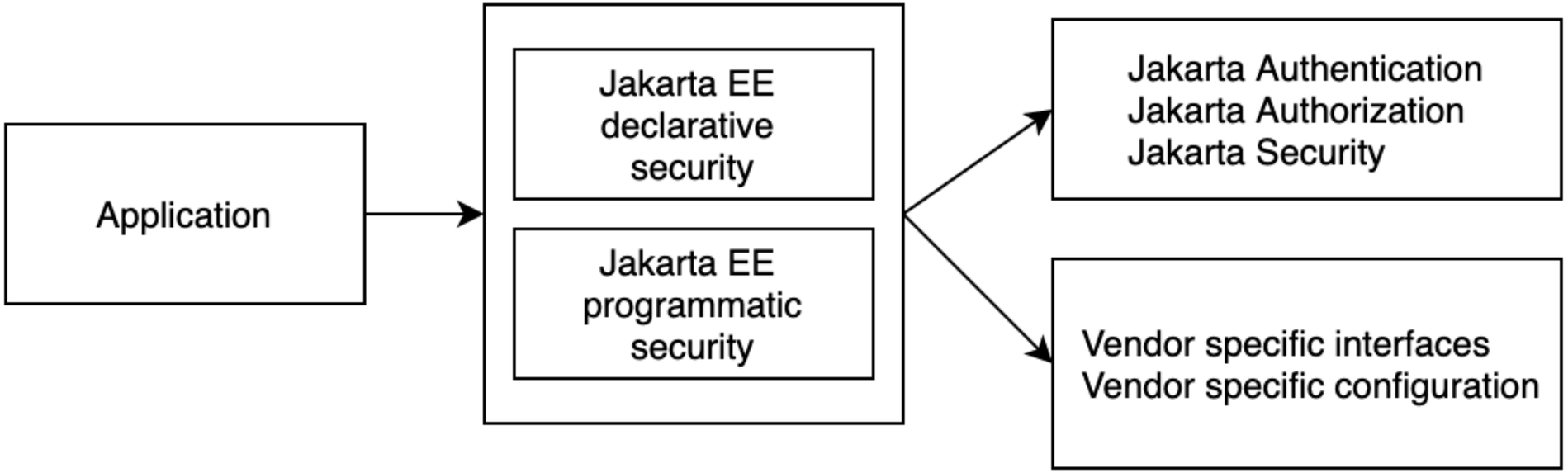
- **SIMPLIFY** security programming model
- Enable **DEVELOPERS** to manage security
- Layered APIs **DELEGATE** to others
- Use **CDI** where appropriate

JAKARTA EE

Application Security

Declarative vs. Programmatic

Jakarta EE supports configuration of an application either using standard APIs or those specific to a server or runtime



Jakarta Authentication

Authentication Mechanism

- Portable API for Authentication
  - Abstracts the specific Identity Store against which to Authenticate
- Simple configuration
- Extensible to support protocols like OAuth / OpenID Connect
- Produces a Consistent representation of an authenticated Subject
- Authentication Events
- Evolution of JASPIC (JSR 196)

OAuth

History

OAuth is a protocol to delegate rights for an application to act on behalf of a user who granted its rights without giving away their login / password

Developed by Twitter, Magnolia and Google,
it was made standard by IETF in April 2010 under RFC 5849

Version 2.0, simpler to use but often criticized by its too many implementations was standardized in October 2012 under RFC 6749 and 6750. It’s already used by many actors (Social Networks like Facebook, Google, Microsoft as well as most API providers)

OAuth

Overview

- An Authorization/Delegation Framework
- Standardized by RFC6749
  - RFC 6750 using bearer tokens
  - RFC 6819 Security considerations
- On a foundation of Token standards
  - JSON Object Signing Encryption (JOSE)
  - JWT (RFC7519), JWS (RFC7515), JWE (RFC7516), JWA (RFC7518), JWK (RFC7517)

OAuth

Concepts

- Actors
  - Resource Owner
  - Client
  - Resource, Resource server
  - Authorization Server
- Authorizations represented as ‘scopes’

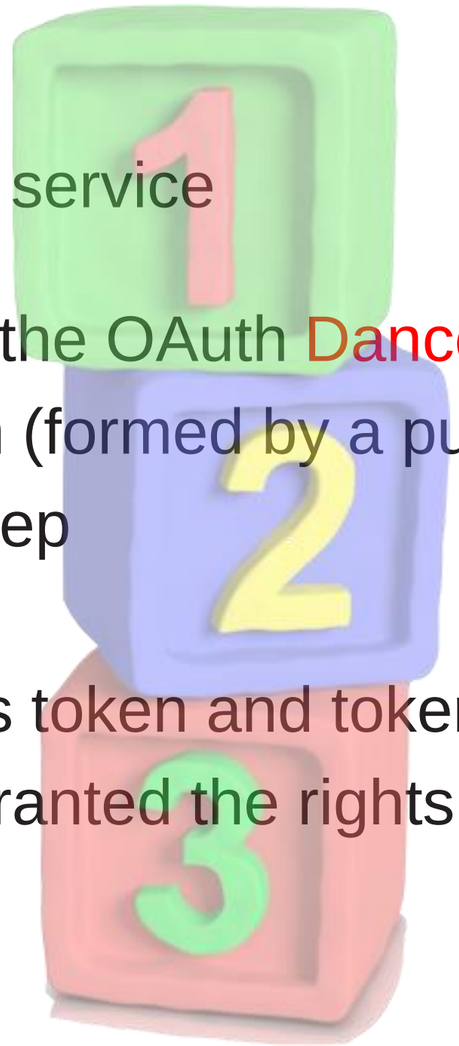
OAuth Dance

OAuth has 3 steps

Creating an application in the OAuth service

Initialization: the right granting phase also called the OAuth Dance. At the end of the dance we obtain an access token (formed by a public and secret part) for the next step

Signature: each request is signed with access token and token identifying the OAuth application that was granted the rights



OpenID Connect (OIDC)

Overview

- Authentication Protocol built on OAuth2
- Session Management – Single Sign on, Out
- An additional Token Type – ID Token
- UserInfo, Discovery, Client Self-registration Endpoints
- Specs : OpenID core, Discovery, Client Registration

OpenID Connect

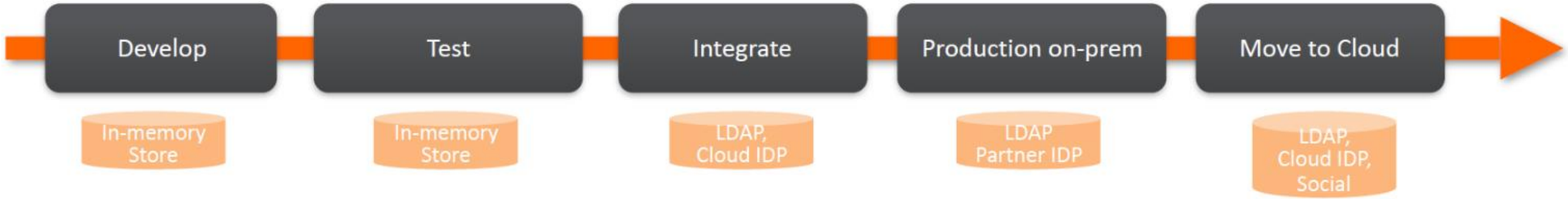
Use Case

- At deployment, Application is configured to be secured by OIDC
- Application must continue to rely on well known abstractions for
  - Identity
  - Authentication
  - Authentication Events

OpenID Connect

What does this mean to the App?

- An App developer
  - Needs a consistent API to abstract the Identity store, authentication mechanism, identity representation
  - Can rely on configuration alone, to change as the App progresses
- DevOps can easily change configuration to suit the environment



Jakarta Authorization

Low-level SPI Authorization Modules

- SPI for Authorization Policy
- SPI for Policy Configuration
- Factory to create and retrieve Policy Configurations
- SPI for Policy Context

Authorization Queries

Testing for Access

```
Subject subject = (Subject)
  PolicyContext.getContext("javax.security.auth.Subject.container");

boolean hasAccess = Policy.getPolicy().implies(
    new ProtectionDomain(
        new CodeSource(null, (Certificate[]) null),
        null, null,
        subject.getPrincipals().toArray(new
  Principal[subject.getPrincipals().size()])
    ),
    new WebResourcePermission("/protected/Servlet", "GET"));
```

Jakarta Security

Creating Secure Applications

- Standardize Terminology
- API for Authentication mechanism
- API for Identity Store
- API for Security Context
- API for Role/Permission Assignment

Security – Identity Store

Overview

- Abstract the Identity Store used by an application
- Simple configuration
- Support a variety of Identity stores
  - Lightweight k-v development stores
  - Traditional stores – LDAP, DB
  - Cloud-specific stores e.g. Social Logins, 3rd-party Cloud Identity providers

Security – Identity Store

Features

- Orderable to support multiple identity stores
- Abstraction to support variety of credential types like
  - Username/Password
  - OAuth Client ID & Secret or JWT Tokens
- Consistent API regardless of container
- Enables Application to determine
  - User’s identity
  - Identity Provider that was used to establish identity
  - Which groups or roles the user belongs to

Security – Context

Definition

```
// Security Context
public interface SecurityContext{
  Principal getCallerPrincipal();
  <T extends Principal> Set<T> getPrincipalsByType(Class<T> pType);
  boolean isCallerInRole(String role);
  boolean hasAccessToWebResource(String resource, String... methods);
  AuthenticationStatus authenticate(HttpServletRequest request,
      HttpServletResponse response,
      AuthenticationParameters parameters);
}
```

Security – Context

Testing for Access

```
// Consider the following Servlet definition
@WebServlet("/protectedServlet")
@ServletSecurity(@HttpConstraint(rolesAllowed = "foo"))
public class ProtectedServlet extends HttpServlet { ... }

// And the following call to hasAccessToWebResource()
securityContext.hasAccessToWebResource("/protectedServlet", GET)
```

Returns true only if the caller is in role "foo".

Security in Jakarta EE 10

JAKARTA EE

Jakarta Authorization 2.1

JAKARTA EE

Cloud Deployments

- Register policy provider programmatically

JAKARTA EE

Jakarta Authentication 3.0

JAKARTA EE

Profiles

- Servlet Container Lite profile
- REST profile
- SOAP profile –> stable

JAKARTA EE

Jakarta Security 3.0

JAKARTA EE

Additional Authentication Mechanisms

- Client-cert and Digest
- OpenID Connect
- Oauth2

Extended Authentication Mechanisms

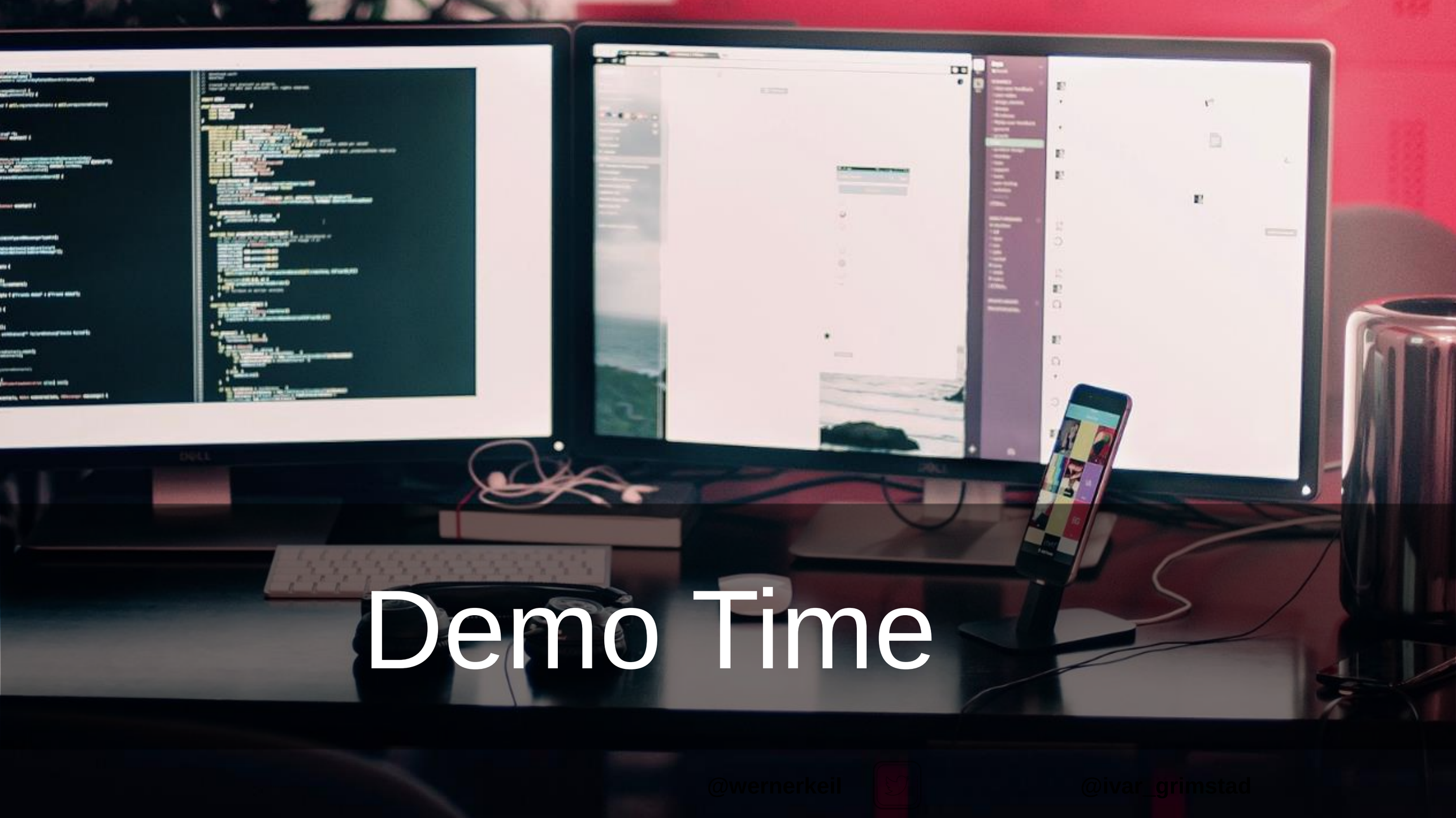
- Authentication Mechanism per URL
- User Choice of Authentication Mechanism
- Multiple Authentication Mechanisms (fallback)

JAKARTA EE

Other

- Additional CDI Support
  - `@RolesAllowed` alternative
- Authorization Modules

JAKARTA EE



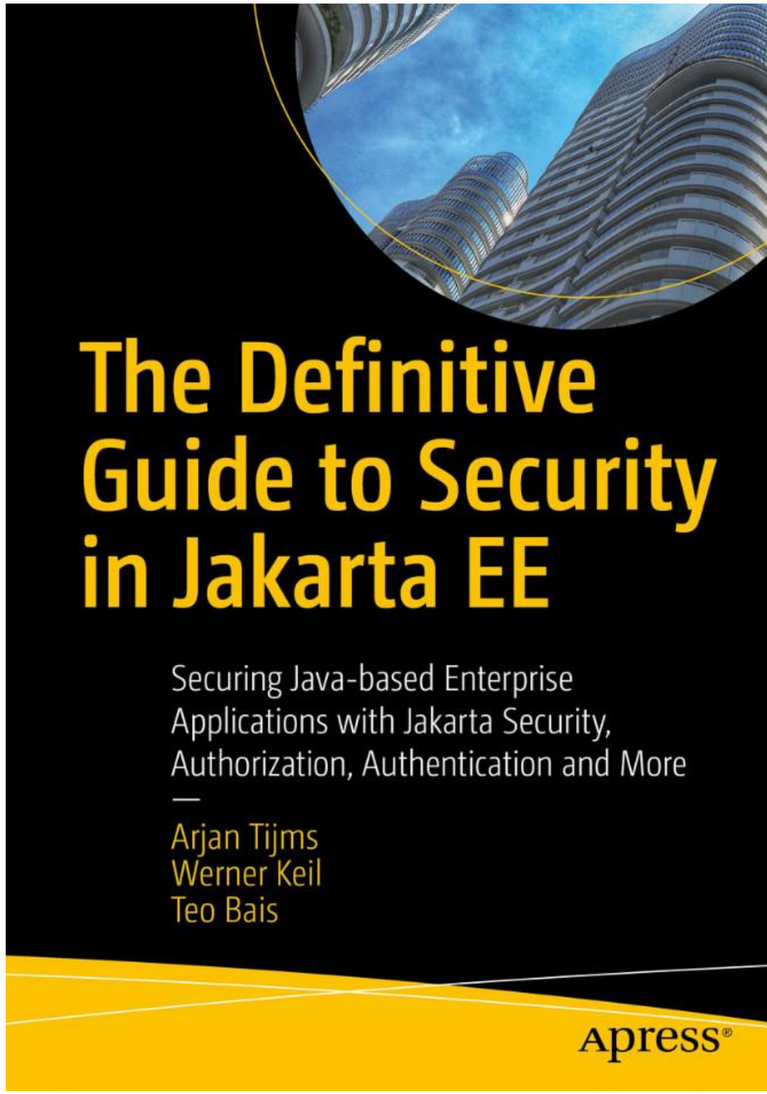
Demo Time

Jakarta Security Book

We (Arjan, Teo and Werner) just finished writing a book
“The Definitive Guide to Security in Jakarta EE (Apress, 2022)

Examples on GitHub:
github.com/Apress/definitive-guide-jakarta-ee-security

Twitter Account:
@jakartasecbook



Resources

- https://github.com/ivargrimstad/security-samples
- https://jakarta.ee/
- https://jakarta.ee/specifications/authentication/3.0/
- https://jakarta.ee/specifications/authorization/2.1/
- https://jakarta.ee/specifications/security/3.0/

JAKARTA EE

JavaLand

March 15 - 17, 2022

in Phantasialand
near Cologne

Q&A

www.javaland.eu

JavaLand

March 15 - 17, 2022

in Phantasialand
near Cologne

Thank You

www.javaland.eu