School of Computer Science
Language Technologies Institute

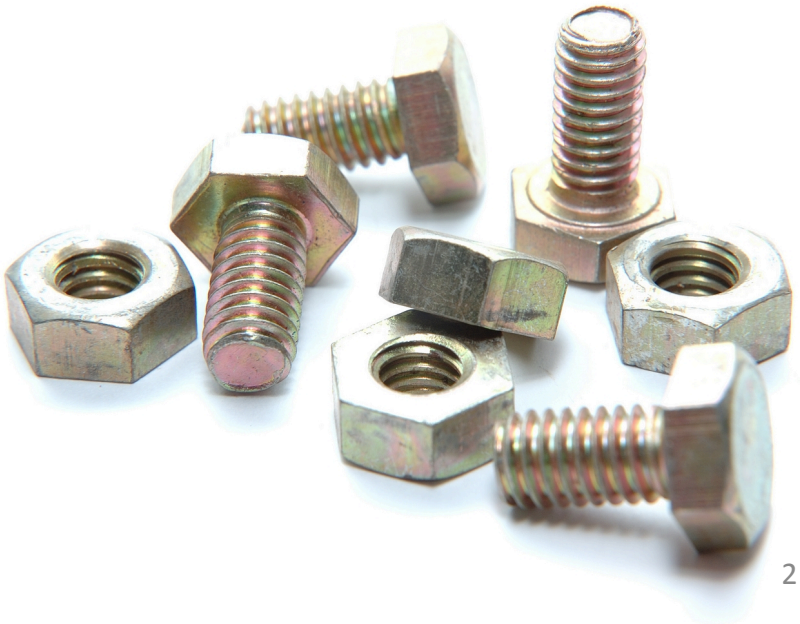
DiscourseDB

Technology and Infrastructure

DiscourseDB Technology

- Hibernate ORM 4
- Spring Platform
  - Spring Data JPA
  - Spring Data REST
  - Spring HATEOAS
  - Spring Boot
- QueryDSL

- Java 8
- Maven 3
- Jenkins
- Artifactory



Hibernate ORM

- Domain model persistence for relational databases
- Database system independent
- Solves Object-Relational Impedance Mismatch

➔ Define entity classes that follow object-oriented idioms including inheritance, polymorphism, association and composition.

Hibernate ORM

```java
@Entity
@Table(name="EMPLOYEE")
public class Employee {

    @Id
    @GeneratedValue
    @Column(name="employee_id")
    private Long employeeId;

    @Column(name="firstname")
    private String firstname;

    @Column(name="lastname")
    private String lastname;

    @Column(name="birth_date")
    private Date birthDate;

    @Column(name="cell_phone")
    private String cellphone;

    @ManyToOne
    @JoinColumn(name="department_id")
    private Department department;

    public Employee() {

    }

    public Employee(String firstname, String lastname, String phone) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthDate = new Date(System.currentTimeMillis());
        this.cellphone = phone;
    }

    // Getter and Setter methods
}
```

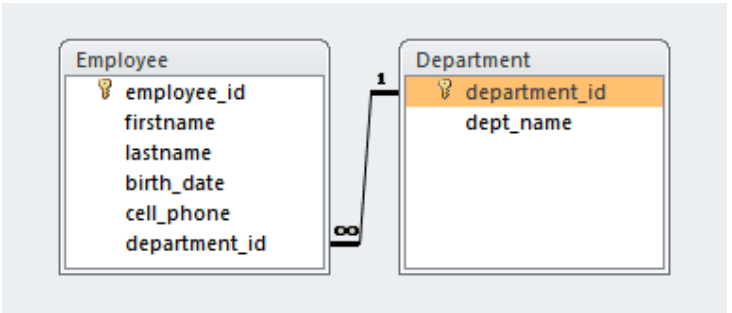
```java
@Entity
@Table(name="DEPARTMENT")
public class Department {

    @Id
    @GeneratedValue
    @Column(name="DEPARTMENT_ID")
    private Long departmentId;

    @Column(name="DEPT_NAME")
    private String departmentName;

    @OneToMany(mappedBy="department")
    private Set<Employee> employees;

    // Getter and Setter methods
}
```



HIBERNATE

Spring Platform

- Easy configuration of data sources
- Session management
- Transaction management
- Dependency Injection: framework provides object instances
- Takes care of all the cross-application layer plumbing in MVC applications

Spring Boot

- Framework for auto-bootstrapping Spring application
- Automatically configures Spring whenever possible based on what you provide in the classpath
- Create stand-alone Spring applications using embedded Tomcat, Jetty or Undertow
- Command Line Apps, RESTful APIs, Web Applications

Spring Data JPA

- Reduce required boilerplate code involved in implementing the data access layer.
- Developer writes repository interfaces, Spring Data automatically creates the implementation

```
public interface UserRepository extends JpaRepository<User, Long> {

  List<User> findByLastname(String lastname);

  User findByEmailAddress(String emailAddress);

  @Query("select u from User u where u.firstname like %?1")
  List<User> findByFirstnameEndsWith(String firstname);
}
```

QueryDSL

- Type-safe queries
- DB independent
- No error-prone query building by string concatenation
- Spring Data integration for easy-to-use joins across repositories

Order

```
List<Person> persons = queryFactory.selectFrom(person)
  .orderBy(person.lastName.asc(),
           person.firstName.desc())
  .fetch();
```

Subqueries

```
List<Person> persons = queryFactory.selectFrom(person)
  .where(person.children.size().eq(
    JPAExpressions.select(parent.children.size().max())
                  .from(parent)))
  .fetch();
```

Tuple projection

```
List<Tuple> tuples = queryFactory.select(
    person.lastName, person.firstName, person.yearOfBirth)
  .from(person)
  .fetch();
```

...

Spring Data REST

- Easily expose Spring Data Repositories via a RESTful service.
- Define REST endpoints for Repositories to expose operations via a web service
  - Update entities using PUT/PATH
  - Delete entities using DELETE
  - Manage entity relationships using POST, PUT, DELETE
  - Search through Repository query methods using GET
  - Sorting and Paging

```
@RepositoryRestResource(collectionResourceRel = "people", path = "people")
public interface PersonRepository extends PagingAndSortingRepository<Person, Long> {
      List<Person> findByLastName(@Param("name") String name);
}
```

Spring HATEOAS

Hypermedia as the Engine of Application State

- Constraint of the REST architecture for building hypermedia driven web services
- Hypermedia is used to navigate the API
- Single (or few) entry points
- Client needs no prior knowledge about interface besides a basic understanding of hypermedia
- Decoupling of Client/Server → independent development

```
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```

```
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
</account>
```

Hypertext As The Engine Of Application State = the API is telling the client what options are available for a particular entity

Infrastructure

Jenkins

artifactory

git

Build System
Project and Dependency Management

+

Continuous Integration

+

Artifact Repository Manager

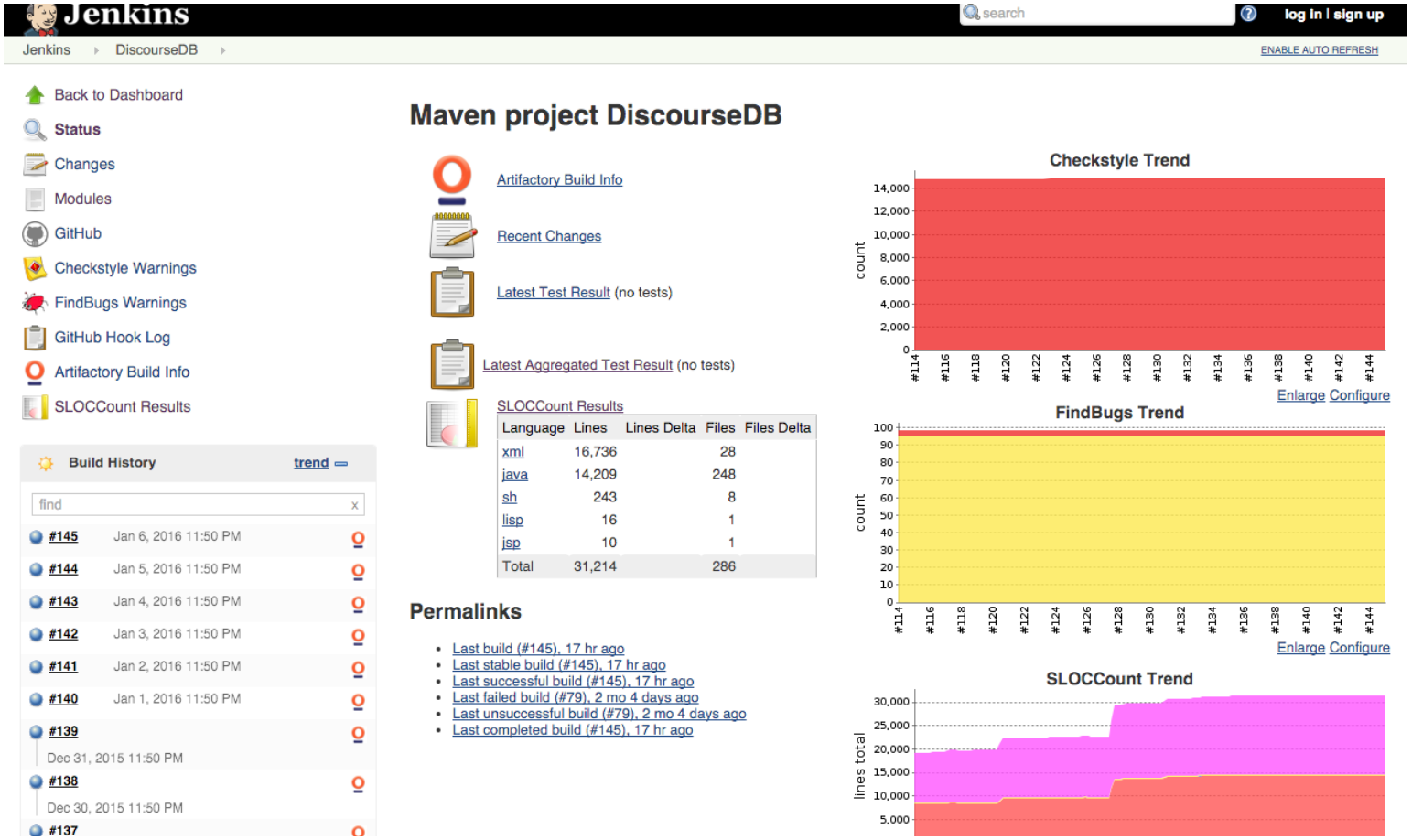
+

Version Control

=

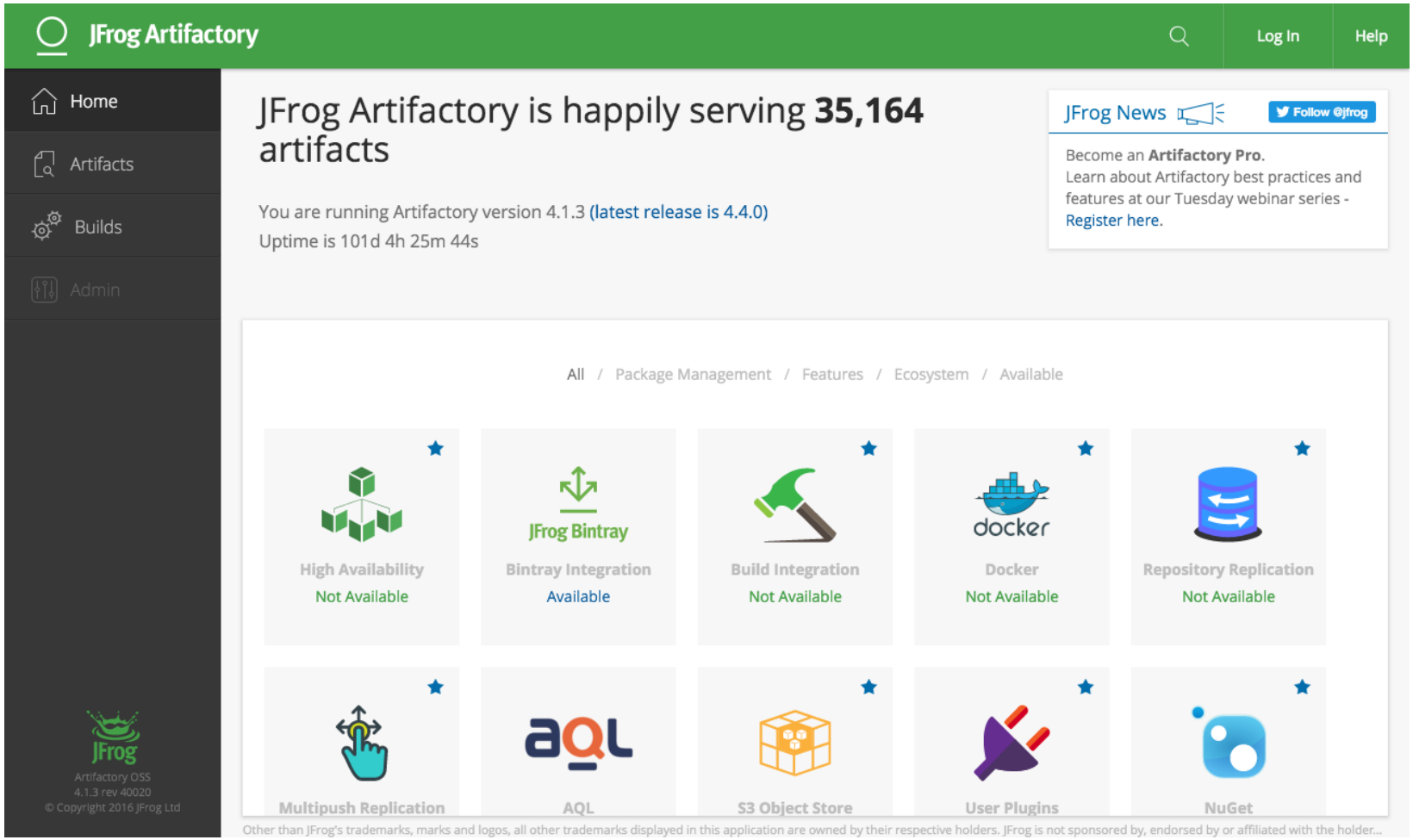
Continuous Delivery

Continuous Integration



http://moon.lti.cs.cmu.edu:8080

Continuous Integration



http://moon.lti.cs.cmu.edu:8081

Questions

- (more about the infrastructure later)