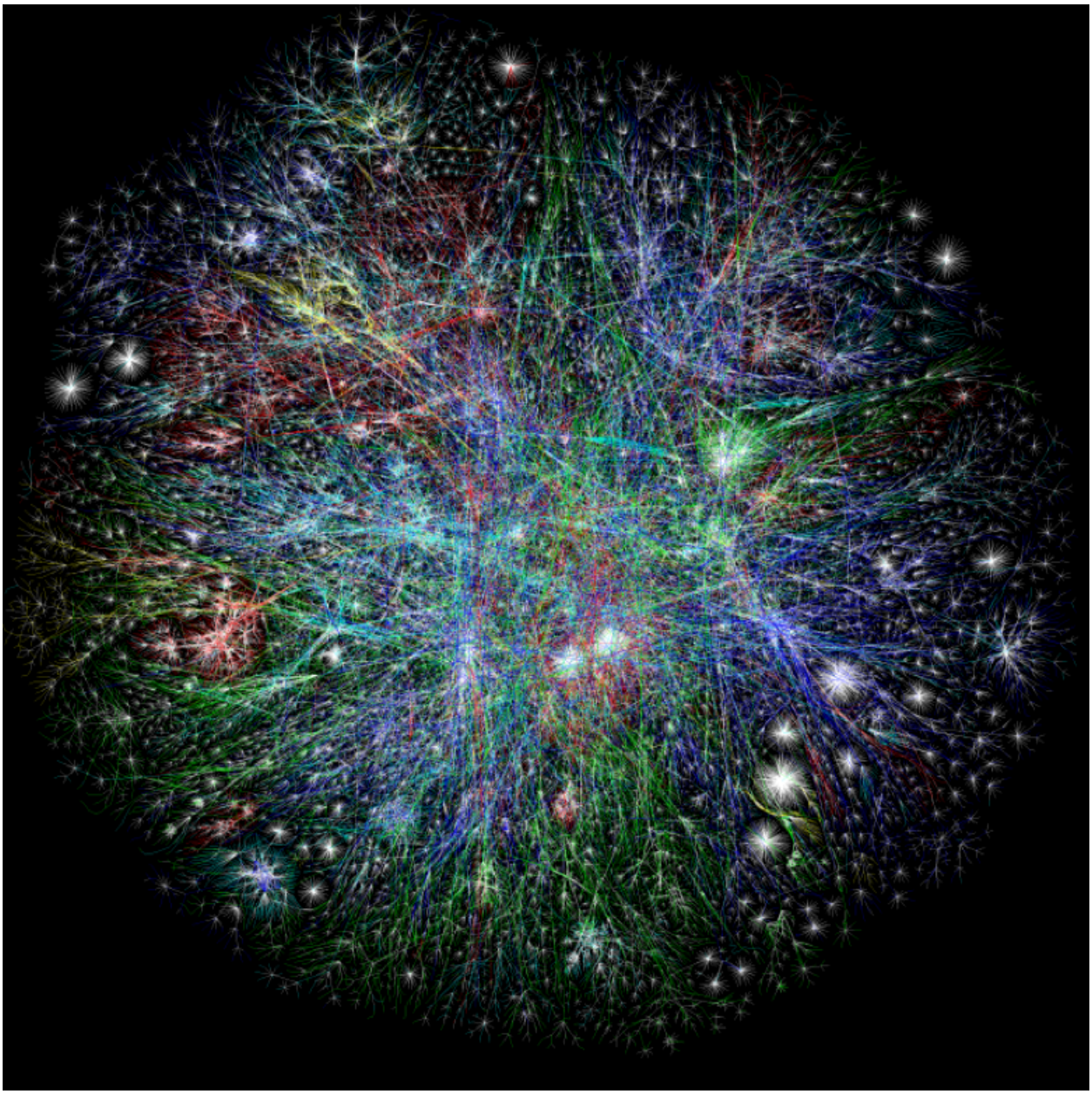
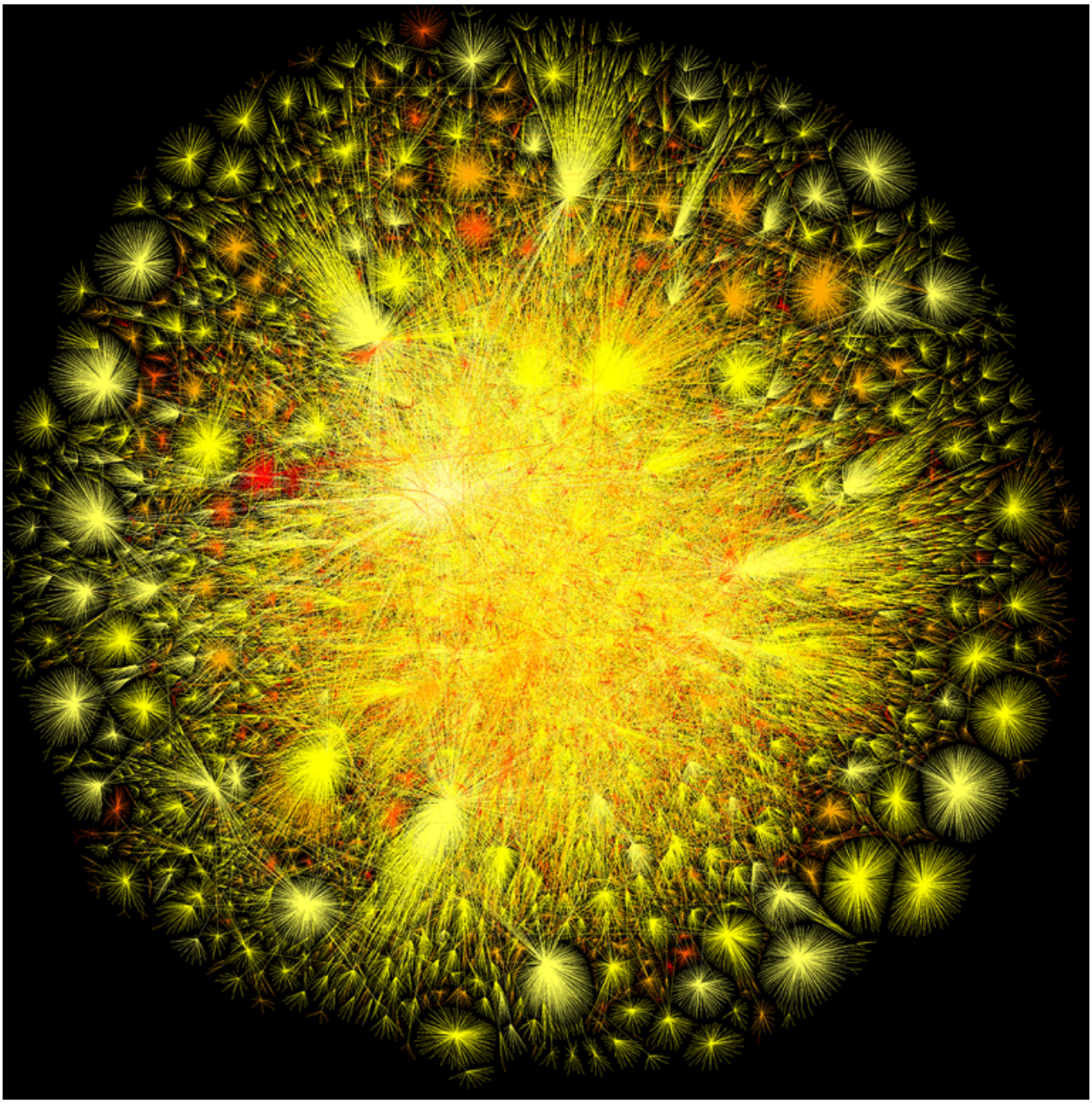
Taller de Introducción a Heroku

Por Luis Daniel Benavides

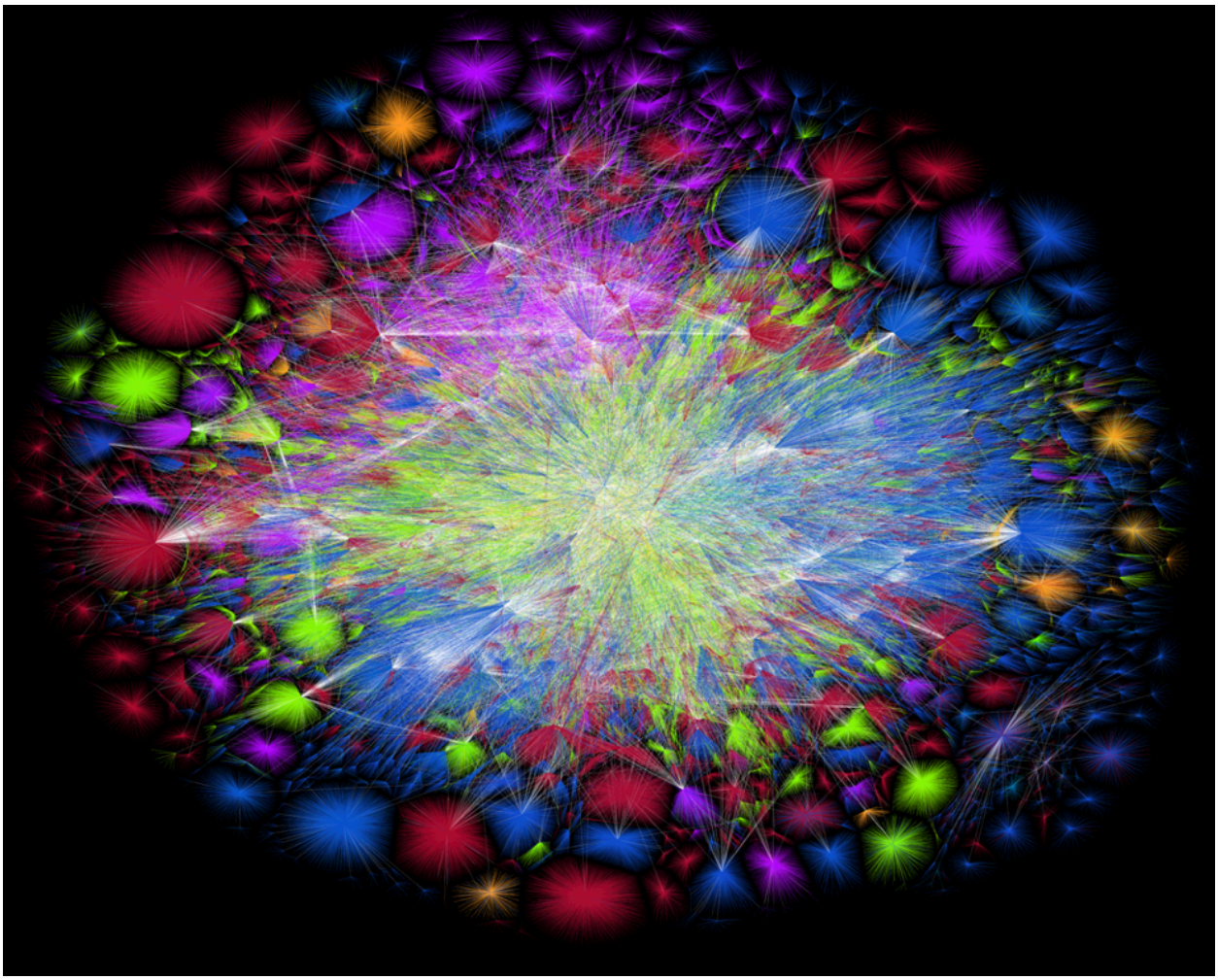
Internet 2003



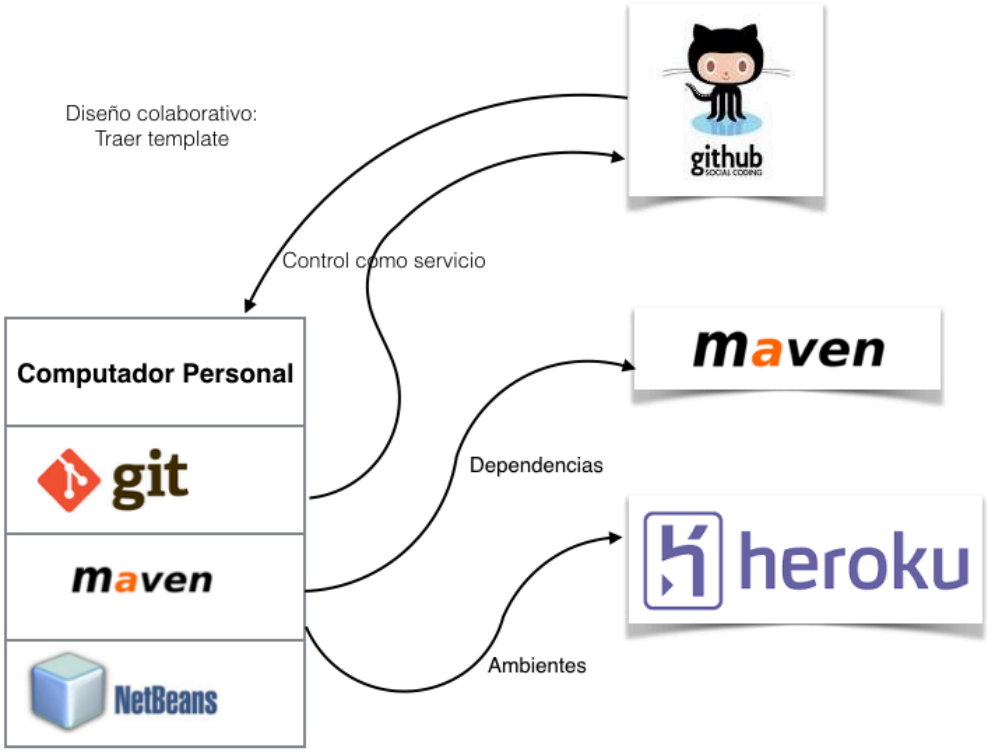
Internet 2010



Internet 2015



Desarrollando en la nube



Para iniciar

- Cree una cuenta en Heroku (www.heroku.com)
- Verifique que Java está instalado
  - `java -version`
    - `java version "1.8.0_60"`
    - `Java(TM) SE Runtime Environment (build 1.8.0_60-b27)`
    - `Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)`
- Verifique que Maven está instalado
  - `mvn --version`
    - `Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T11:41:47-05:00)`
    - `Verifique que Git está instalado`
    - `git --version`
    - `git version 2.6.4`

Descargue e instale el Toolbelt de Heroku

- Esta herramienta le permite manipular por medio de comandos la infraestructura en la nube donde desplegará su aplicación

Cree una aplicación Java simple

- Cree una aplicación usando maven y GIT
    - `$ mvn archetype:generate -DgroupId=edu.escuelaing.arem -DartifactId=SparkWebApp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`

- `$ cd SparkWebApp`
- `Cree los archivos recomendados`
    - `$ echo 'Mi primer proyecto' > README.txt`
    - `$ echo 'TODO: Copiar el texto de la licencia http://www.gnu.org/licenses/gpl.html' > LICENSE.txt`
    - `$ echo '# TODO: Copiar los ontenidos de https://github.com/github/gitignore/blob/master/Java.gitignore' > .gitignore`
- `Cree el repositorio local de Git`
    - `$ git init`
- `Adicione los archivos al control de Git`
    - `$ git add *.txt pom.xml .gitignore src`
    - `git status -s`
- `agregue las siguientes líneas a .gitignore`

```
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
```

```
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*

# Ignora todos los archivos en el directorio target en los proyectos creados con maven
target/

# Ignora archivo .DS_Store en Mac
.DS_Store
```

- git status -s
- agregue .gitignore nuevamente al area staged
  - $ git add .gitignore
- Cree la primera versión de su proyecto
  - $ git commit -m ‘Primera versión de proyecto XXXX de Nombre'
- **Cree un repositorio en GitHub**
- Adicione al repositorio remoto
  - git remote add origin https://github.com/dnielben/xxxxxxx.git
- Revise que repositorios remotos están configurados
  - git remote
- Empuje su proyecto y haga que Git rastree los cambios en el repositorio remoto
  - git push -u origin master

Construya una aplicación Web simple usando SparkWeb

Cree una clase principal Java similar a esta:

```
package co.edu.escuelaing.arem.designprimer;

import static spark.Spark.*;

public class SparkWebApp {
```

```
    public static void main(String[] args) {
        get("/hello", (req, res) -> "Hello Heroku");
    }
}
```

Agregue las dependencias al POM

Esto trae la librería de spark desde el repositorio de Maven

```
...
    <properties>
        ...
    </properties>
    <!-- library dependencies -->
    <dependencies>
        <dependency>
            <groupId>com.sparkjava</groupId>
            <artifactId>spark-core</artifactId>
            <version>2.7.2</version>
        </dependency>
    </dependencies>
    ...
</project>
```

Ejecute la clase

Su aplicación debe estar disponible en

- http://localhost:4567/hello
- Spark es un framework para crear aplicaciones web rápidamente.
- ¿Puede describir la arquitectura de Spark?
    - http://sparkjava.com/

Ahora preparemos la aplicación para ser desplegada en Heroku

Modifique la aplicación para leer el puerto desde el entorno del sistema operativo. Heroku provee un puerto determinado usando la variable de entorno “PORT” (Aquí está usando nombres para desacoplar).

```
package co.edu.escuelaing.arem.designprimer;

import static spark.Spark.*;
```

```
public class SparkWebApp {

    public static void main(String[] args) {
        port(getPort());
        get("/hello", (req, res) -> "Hello Heroku");
    }

    static int getPort() {
        if (System.getenv("PORT") != null) {
            return Integer.parseInt(System.getenv("PORT"));
        }
        return 4567; //returns default port if heroku-port isn't set
(i.e. on localhost)
    }
}
```

Asegúrese que Maven copia las dependencias al directorio target

Modifique le POM.xml para que contenga los plugins de compilación y de dependencias. Esto asegura que se compile a la versión 8 de Java y que las dependencias se copien en el directorio target,

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apa
che.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>co.edu.escuelaing.arem.designprimer</groupId>
    <artifactId>01TallerSparkMavenGitHeroku</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <!-- library dependencies -->
    <dependencies>
        <dependency>
            <groupId>com.sparkjava</groupId>
            <artifactId>spark-core</artifactId>
            <version>2.7.2</version>
```

```
        </dependency>
    </dependencies>

    <!-- build configuration -->
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <version>3.0.1</version>
                <executions>
                    <execution>
                        <id>copy-dependencies</id>
                        <phase>package</phase>
                        <goals><goal>copy-dependencies</goal></goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

El Procfile

La procfile es un archivo en su directorio que indica que comando se debe ejecutar para iniciar la aplicación en Heroku

- Cree un archivo denominado Procfile en la raíz de su proyecto y agréguele lo siguiente
    - `web: java $JAVA_OPTS -cp target/classes:target/dependency/* co.edu.escuelaing.designprimer.SparkWebApp`
- La palabra web indica que heroku le enviará tráfico web cuando se despliegue

- El resto es el comando Java para ejecutarlo
- **Agregue el archivo a su proyecto en git**

Construya y corra la aplicación localmente emulando Heroku

```
$ mvn clean install
```

```
$ heroku local web
```

Para abrirla puede ir a http://localhost:5000/.

Despliegue su aplicación en Heroku

- Verifique que todo este incluido en GIT

```
$ git add .
$ git commit -m “Proyecto completo."
$ heroku login
Enter your Heroku credentials.
...
$ heroku create
Creating arcane-lowlands-8408... done, stack is cedar-14
http://arcane-lowlands-8408.herokuapp.com/ | git@heroku.com:arcane-
lowlands-8408.git
Git remote heroku added
$ git remote
$ git push heroku master
...
-----> Java app detected
...
-----> Launching... done
       http://arcane-lowlands-8408.herokuapp.com deployed to Heroku
```

- Para abrir la aplicación escriba
  - heroku open.