# GlassCat

*The JavaScript Application Server*

Last update: 12/17/2017

# Description

*"An application server is a software framework that provides both facilities to create Web applications and a server environment to run them"* [1].

**GlassCat is:**

- a JavaScript Application Server
- the default JEC implementation
- built over NodeJS
- written in TypeScript

**GlassCat as:**

- a standard for JavaScript Web apps
- a portable development environment
- an efficient alternative to Spring Boot and Java EE
- the best integrated solution for TypeScript development

[1] Wikipedia: Application server

# Positioning

GlassCat is an alternative to JAVA technologies and improves the development of NodeJS applications.

Benefits vs. Spring Boot / Java EE[**]:

- non-blocking architecture
- stateless support for REST APIs[**]
- uniform programmatic language
- short learning curve

Benefits vs. NodeJS / Express:

- development standardization
- portability / reusability
- built-in security support
- TypeScript integration

Lower infrastructure costs

Better productivity

# JEC Default Implementation

GlassCat is the JEC default implementation.

JEC specifications bring industrial processes to JavaScript applications development:

- scalability and maintainability
- portability
- test driven development
- IoC / Dependency Injection
- container orchestration
- microservice architectures
- RESTful services architectures
- etc.

JEC is an open source specification entirely based on abstraction layers.

# Ease-of-use

JEC ships with a command line tool that can be used if you want to quickly prototype with GlassCat.

It allows you to install and run a GlassCat server without any boilerplate code.

To install JEC-CLI, run the following in a terminal window:

```
$ npm install jec-cli -g
```

Now you can use JEC-CLI from anywhere. The commands below install a GlassCat server in a new directory and run it immediately:

```
$ mkdir test-server
$ cd test-server
$ jec install-glasscat
$ glasscat start
```

# TypeScript Integration

GlassCat is entirely written in TypeScript and supports generic types and annotations:

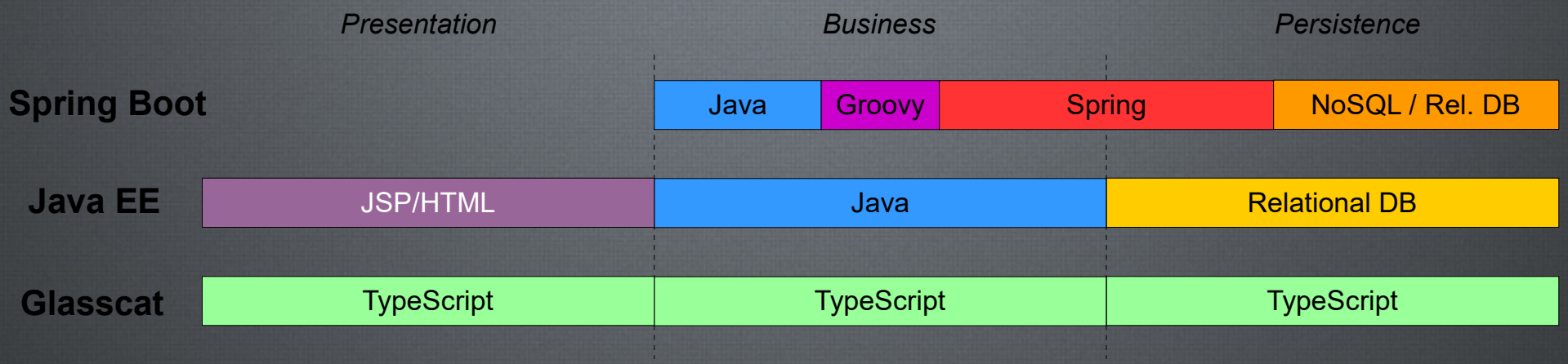
```
@WebJslet({
  name: "HelloWorld",
  urlPatterns: ["/say-hello"]
})
export class HelloWorldSvc extends HttpJslet {

  public doGet(req:HttpRequest, res:HttpResponse, exit:Function):void {
    let message:string = this.builder.sayHello();
    exit(req, res.send(message));
  }

  @Inject()
  private builder:MessageBuilder;
}
```

TypeScript integration highly increases productivity and provides a well structured environment for designing scalable architectures.

# The Unified API

Unified APIs offers strong benefits:

- better maintainability
- lower learning curve cost
- better agility integration
- better information system governance




Glasscat improves the MEAN[1] stack:

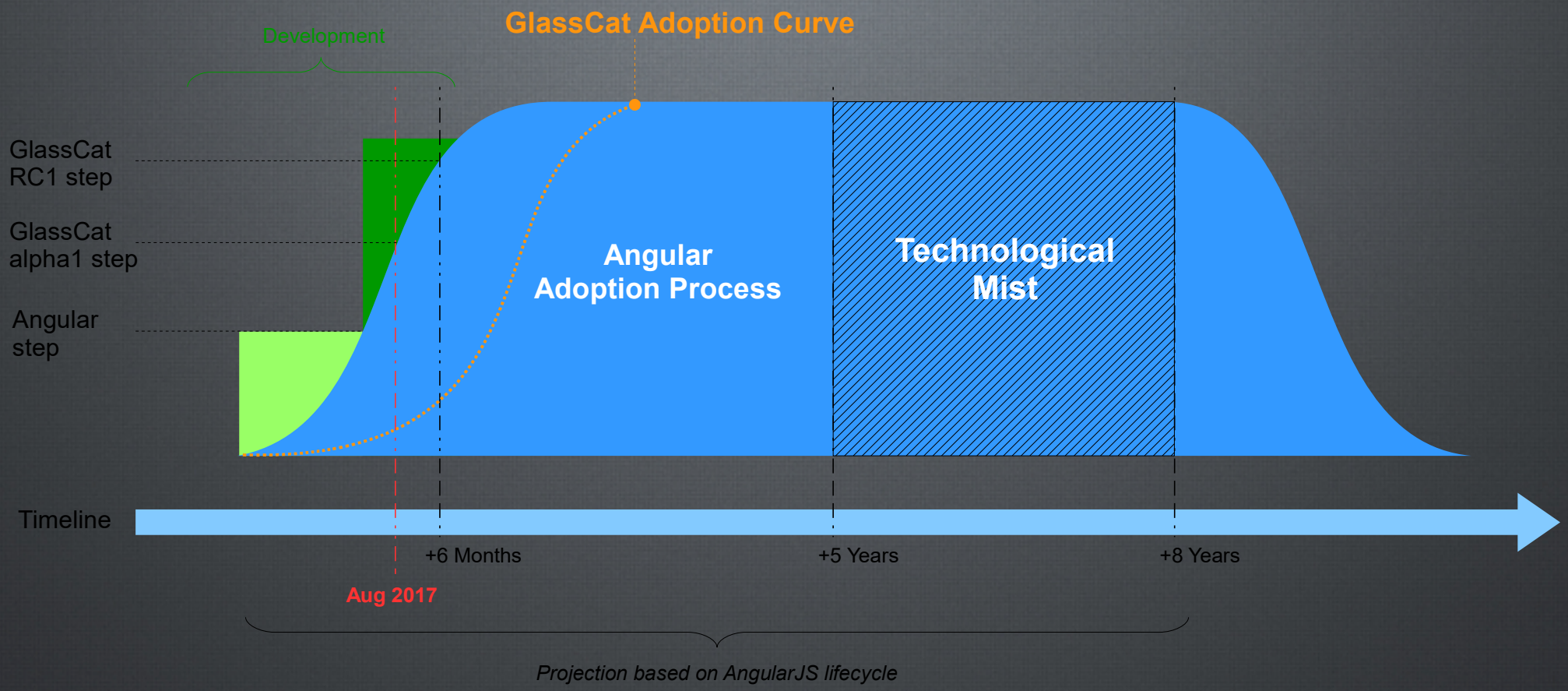
- standardized architecture
- built-in security layer
- built-in development tools
- centralized integration
- packaging support for portability

[1] MongoDB + Express + Angular + Node

# Angular Integration

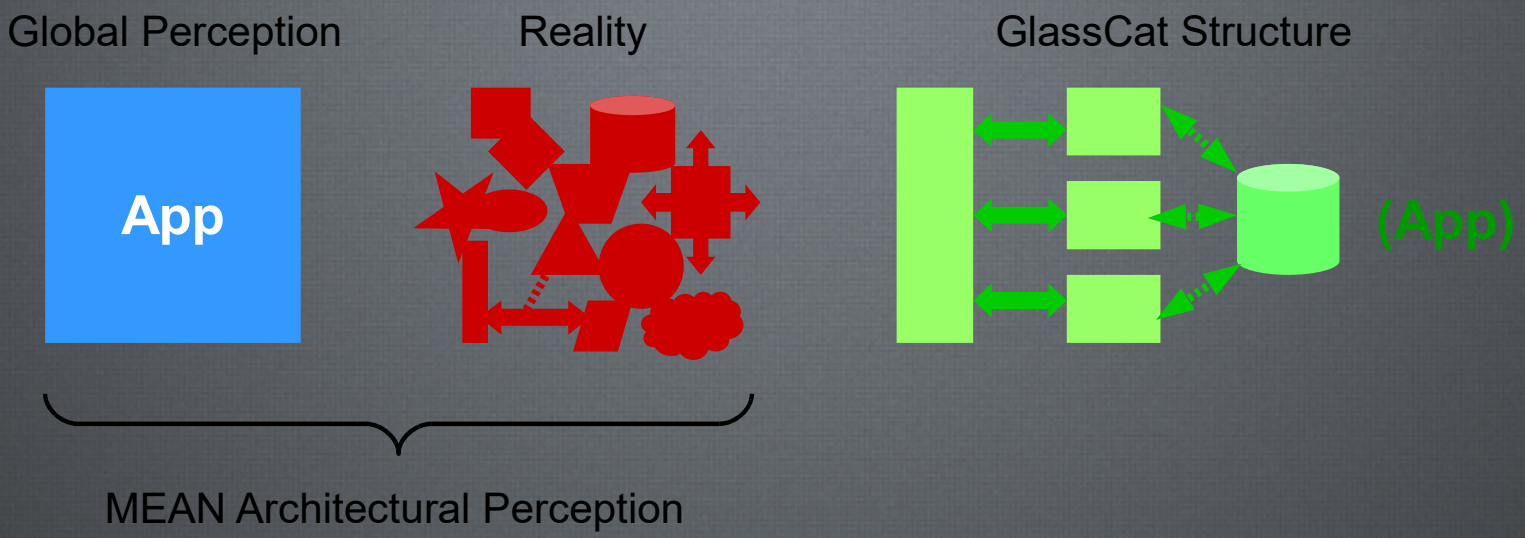
Angular is :

- natively integrated to GlassCat
- a powerful leverage point for GlassCat massive adoption



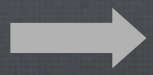
# Solving Real World Issues

Developers have an idealized perception of the MEAN stack:




GlassCat provides business-oriented development structure beyond fad:

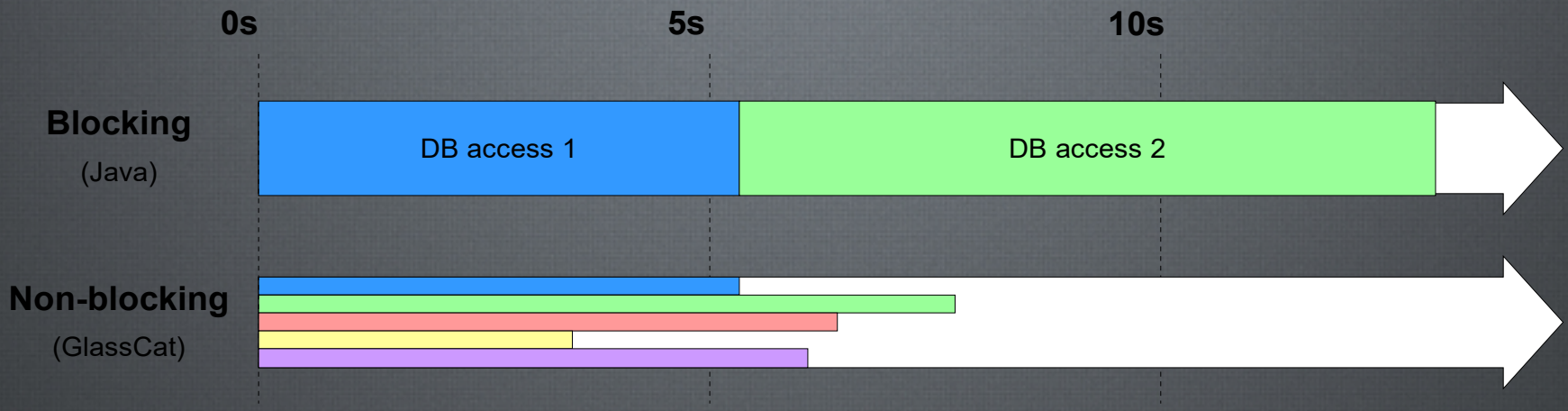
- faster products development
- better maintainability
- long-term vision



**Strong Costs Reduction**

# Non-blocking Architecture

Non-blocking architectures reduce infrastructure costs by mitigating performance issues due to network latency.




GlassCat is built on top of Node.js which enables support for tens of thousands of concurrent connections without incurring the cost of thread context switching[1].

[1] Wikipedia : Node.js

# URL-mapping Architecture

URL-mapping (Jslet) is based upon Servlet 3.0 specification. Jslets have the following advantages over other server extension mechanisms.

They:

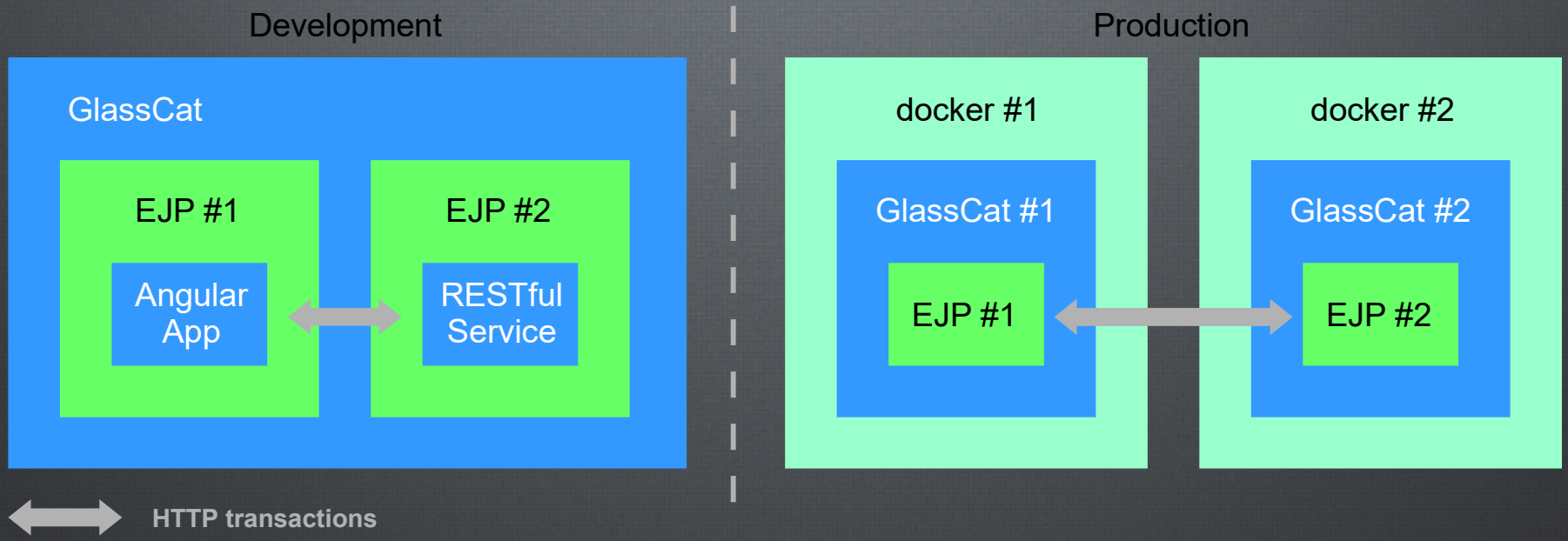
- are faster than other server-side scripts because a different process model is used
- use a standard API
- come with all of the TypeScript advantages (ease-of-development, type checking, ES6/7 support, etc.)
- provide portability and platform independence
- access the large set of APIs available for the JavaScript echosystem
- can be used with the native JEC stateless mode

**The GlassCat stateless architecture is much faster than Java stateful architectures.**

# The Container Approach

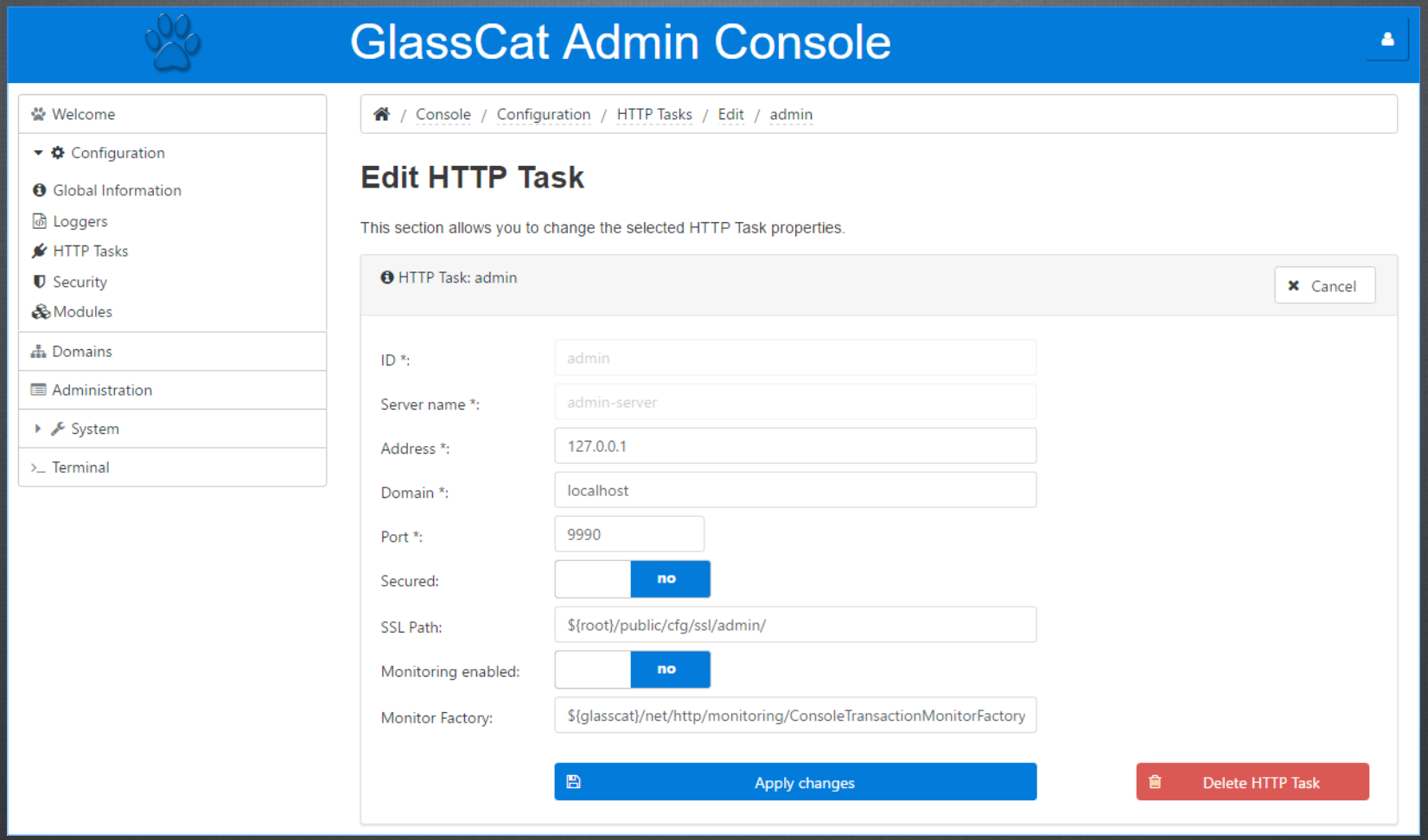
JavaScript containers, as defined by JEC, are micro-architectures oriented:

- JEC project unit is called Entreprise JavaScript Project *(EJP)*
- EJPs can combine business and presentation layers *(deprecated)*
- it is best to have only one service, or application, by EJP



# Administration Console

Administration console is an Angular application available from a standard EJP.



# GlassCat CLI

All actions available in the GlassCat Administration Console can be done by using the Command Line Interface:

```
$ glasscat create-jslet --projectPath=FooBar --name=HelloWorld -patterns="/hello"
```

```
import { HttpJslet, WebJslet, HttpRequest, HttpResponse } from "jec-exchange";
import { HttpStatusCode } from "jec-commons";

@WebJslet({
  name: "HelloWorld",
  urlPatterns: ["/hello"]
})
export class HelloWorld extends HttpJslet {

  public doGet(req:HttpRequest, res:HttpResponse, exit:Function):void {
    // TODO Auto-generated method stub
    exit(req, res.send("Hello World!"), null);
  }
}
```

# JEC Integration

GlassCat ships with the default JEC implementations:

| Specification | Description | Implementation |
|---|---|---|
| Jslet | A jslet is a small JavaScript program that runs within a JEC server. Jslets receive and respond to requests from Web clients across HTTP. | GlassCat |
| JARS | The JavaScript API for RESTful Services *(JARS)* provides functionalities for easily creating RESTful web services. | Sandcat |
| JUTA | The JavaScript Unit Testing API *(JUTA)* allows developers to write portable unit testing based on the same kind of annotations than Junit. | Tiger |
| JDI | The JavaScript Dependency Injection *(JDI)* specification defines the API that provides the ability to inject components into an application in a typesafe way. | Bobcat |

EJPs can directly refer to these implementations without additional imports.

All the comming JEC specifications will be included in GlassCat.

# Integrated Development Solution

GlassCat architecture has been designed to facilitate the server administration by developers:

- special attention has been paid to remove complexity
- workspace is directly integrated to server
- everything can be done from Visual Studio Code Editor (or any other code editor)

Portability means that:

- GlassCat can be deployed over Linux, Windows and Mac OS
- EJPs can easily be packaged to export, or import, projects
- GlassCat has been designed to be deployed through Container Orchestration (startup time, memory footprint...)

GlassCat is great for DevOps and agility!

# Archetypes Manager

GlassCat has its own archetype manager: Wildcat.

Archetype will:

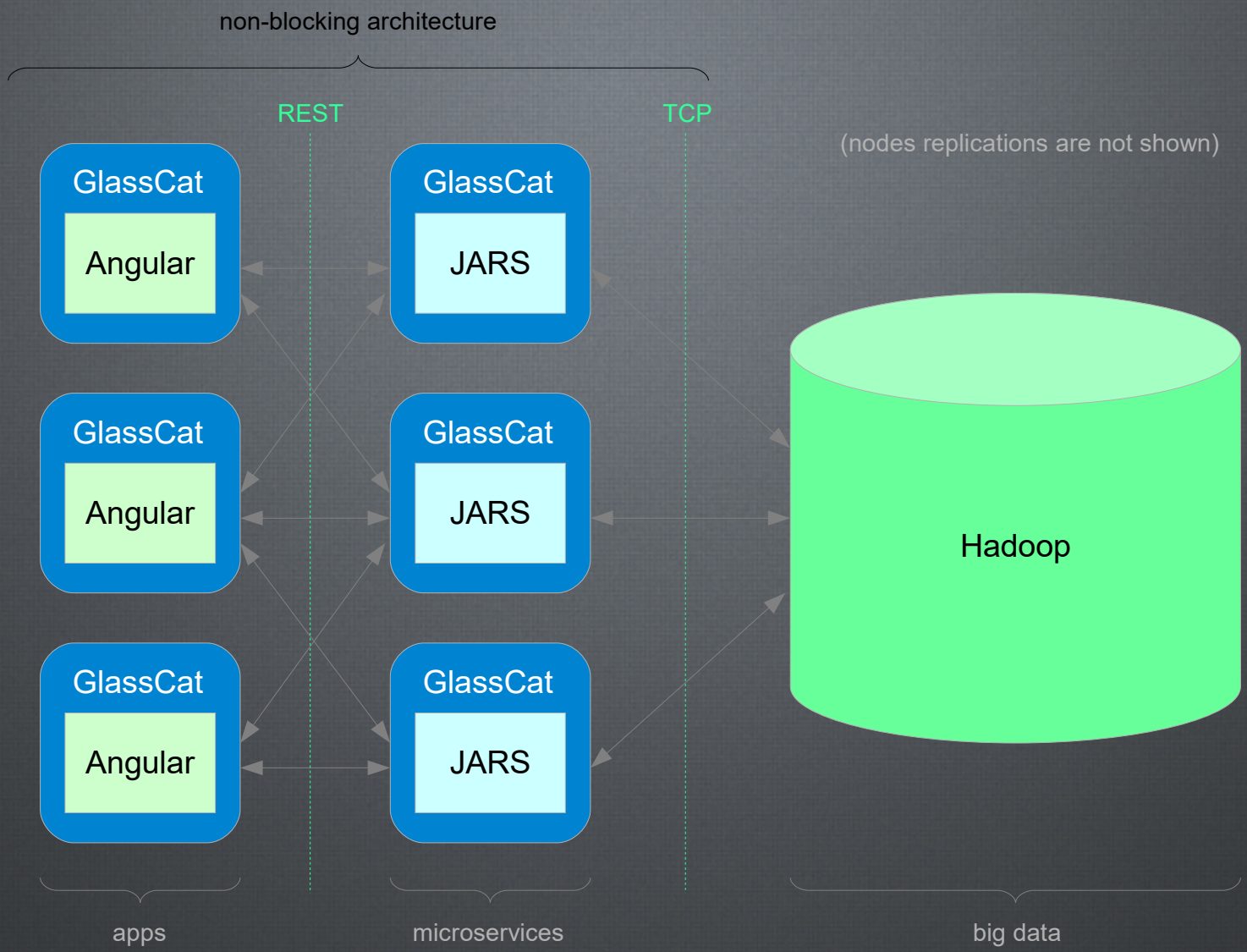
- help authors create JEC project templates for users
- provides users with the means to generate parameterized versions of those project templates

GlassCat provides default archetypes for building:

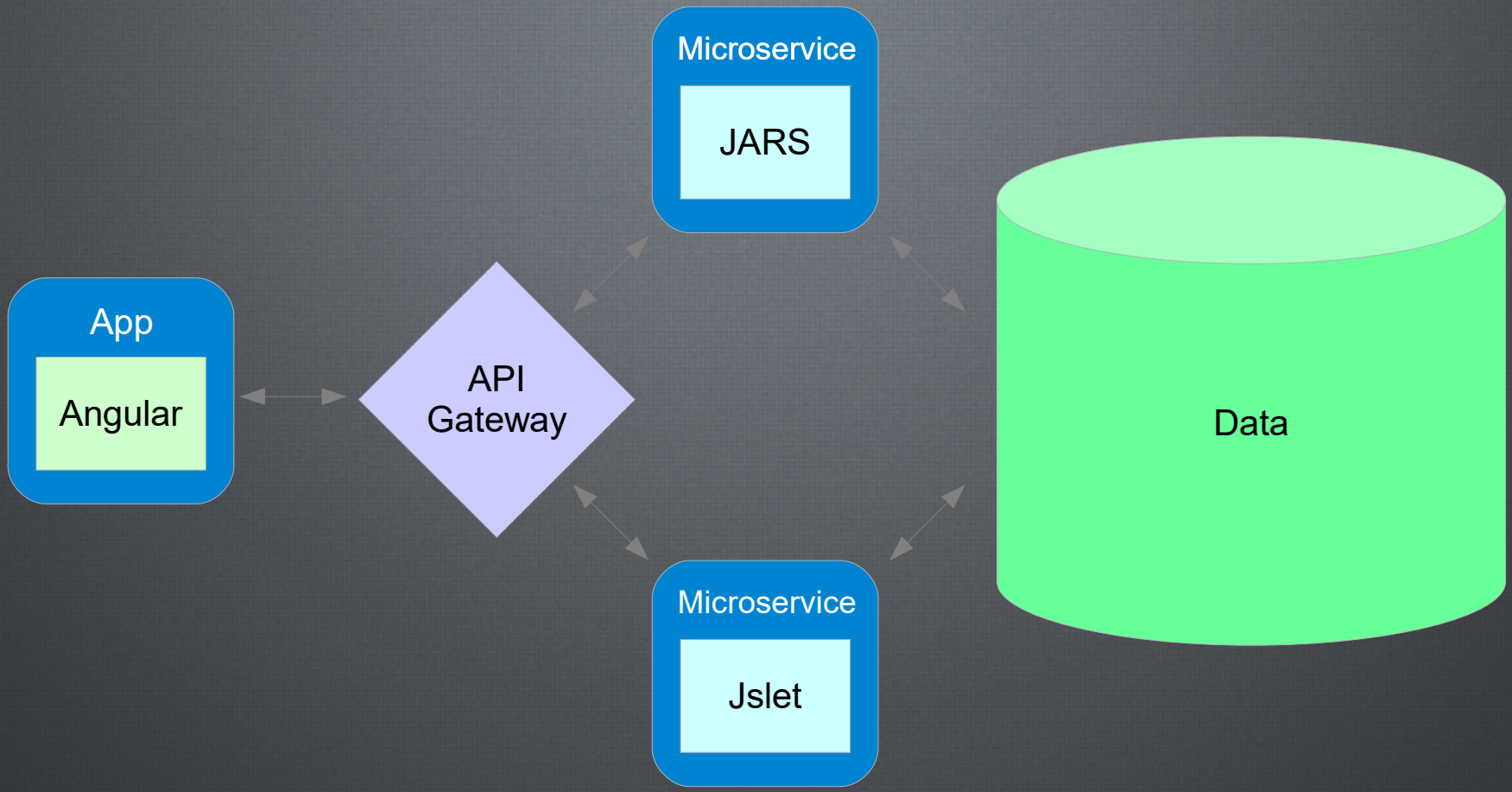
- standard applications based on jslets
- Angular apps
- RESTful-based web services

Authors can create their own archetypes, accessible from any public, or private, repositories.

# Big Data Architecture Sample



# Microservice Architecture Sample



# Additional Features

GlassCat includes the following features:

- server-side customizable templating API
- security layer and Roles support
- integrated session mechanism
- projects import / export mechanism
- i18n support
- built-in transactions analytics
- static resources serving
- CSS inner references support

The features below will be included in the comming releases:

- additional custom server modules
- built-in timer services
- built-in Data Access API
- Helmet[1] security integration

[1] HTTP headers security module

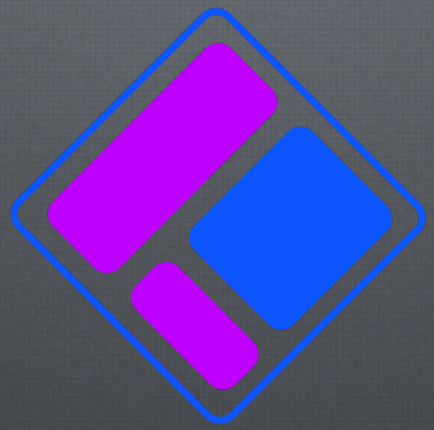
# Where to go from here?

For more information and documentation on GlassCat and JEC please visit:

- GlassCat Project
- JEC Sample Projects
- JEC Youtube Channel

GlassCat is part of the JEC project:

- JEC project on GitHub



JEC

JavaScript Entreprise Container