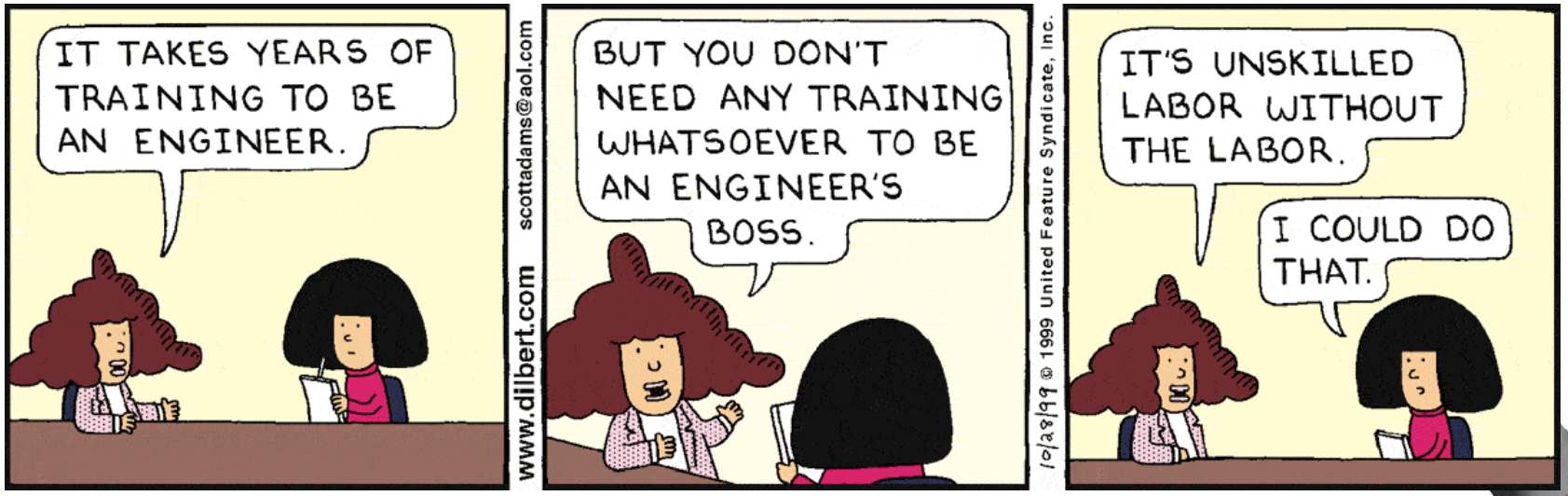
Enterprise Application Development in the Cloud Workshop Training

v0.4

Welcome / Introduction



Workshop Introduction

- The next generation Development Platform for developing Enterprise Applications will likely be browser and most certainly cloud based.
- This Workshop will demonstrate what this Development Platform will look like and give you a hands on opportunity to experience this platform.
- The Development Platform could consist of the following Cloud based components:
  - ★ IDE
  - ★ Development Runtimes
  - ★ Production Runtimes
  - ★ Source Control System
  - ★ Automated DevOps and Build System

Expectations

- ✓ We will meet 1-2 times a week for 1-2 hours.
- ✓ We will meet for 4 weeks.
- ✓ All meetings will be held remotely via Zoom Conferencing.
  - ✳ We will review a Cloud based architecture, technologies, and walk through the code for an IoT Reference Application.
  - ✳ You will be given basic instruction and also the resources (tutorials, web sites, and sample code) so that you can then build your own IoT apps using Cloud technologies and platforms.
  - ✳ You are expected review the workshop materials and build your IoT apps. outside of the workshop.
  - ✳ You are expected to setup your own tools and cloud accounts.
  - ✳ I have setup a Google Group and Padlet for Questions and Collaboration.
  - ✳ We will also use the workshop to review questions and answers.

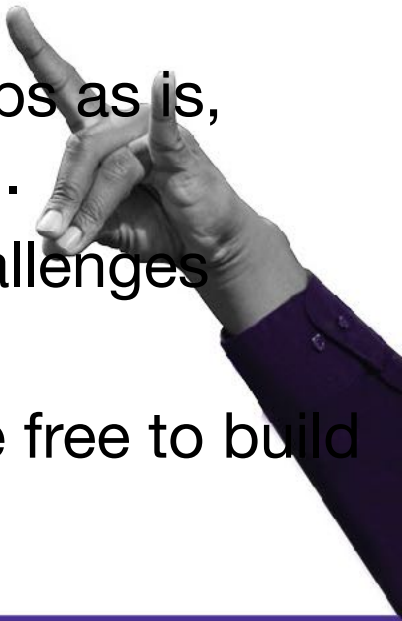
What New Stuff Will You Learn?

Domain	Details
Java	Spring Framework (Core, Security, JDBC) - Alternative to Java EE.
Java	JAX-RS API - To build a REST API based Server App in Java.
Java	Apache Tomcat Java Server / Redhat Wildfly- Java Application Server.
Java	Maven - Build and Dependency Management Tool.
PHP	LavaChart, Plotly, and Guzzle - Libraries to build the Reporting App.
PHP	Composer - Dependency Management Tool.
Cloud	GitHub - Cloud based Version Control System.
Cloud	Codenvy - Cloud based IDE (or you can use Eclipse).
Cloud	OpenShift - PaaS based Cloud.

Use this experience as foundation to build other apps or for your Capstone project!

More Expectations

- You are free to leverage my Raspberry Pi as I can post IoT data to your IoT Services App or you can just use your own Pi.
  - My Raspberry Pi IoT app is designed to connect to a list of backend Cloud Servers however you will need to build your IoT Services app to the specifications outlined in the SDK.
- You are free to build and deploy the Reference apps as is, study the code, as well as modify them as desired.
- You all know Java so you should not have any challenges learning how to build the IoT Services app.
- If you do not know the Laravel Framework you are free to build the IoT Reporting app in plain PHP.

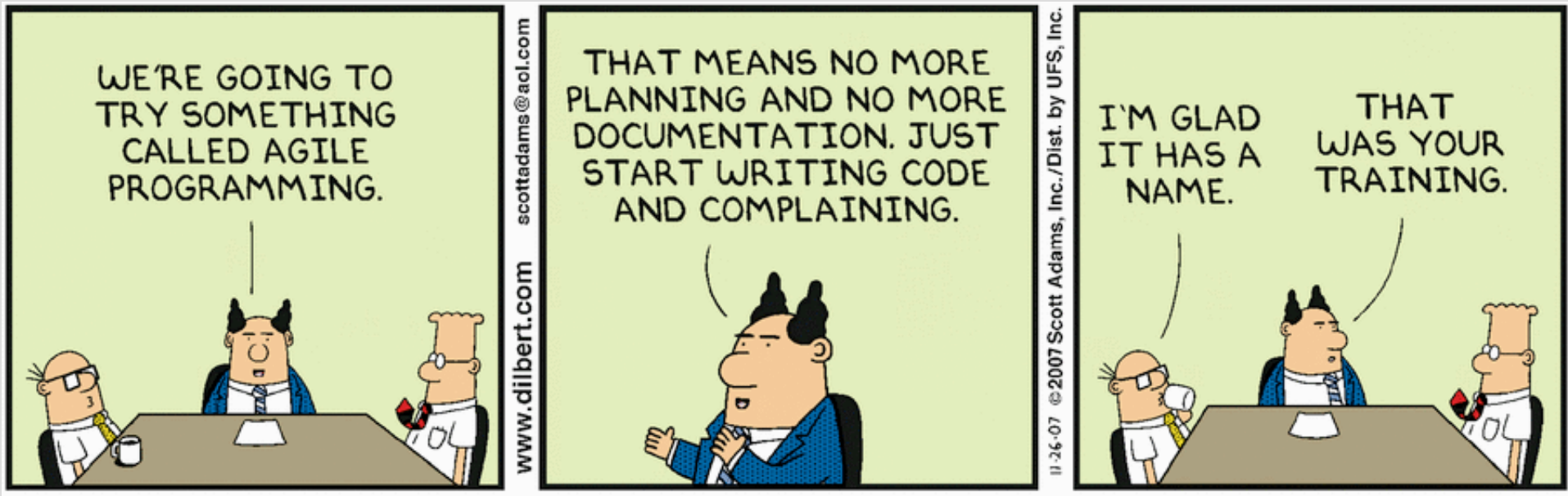


Planned Workshop Sessions

Session	Topic	Week - Day
1	Overview of SDK, Reference Architecture, and the IoT Device App. (Python)	1 - Tues.
2	IoT Services App. (JAX-RS, Spring, Maven)	2 - Tues.
3	Build and deploy the IoT Services App.	2 - Thurs.
4	IoT Reporting App. (PHP, LavaCharts, Guzzle	3 - Tues.
5	Build and deploy the IoT Reporting App.	3 - Thurs.
6	General Q/A and Troubleshooting	4 - Mon.

Questions?
Are You Ready?

Workshop Session 1



Workshop Session 1

- SDK and Reference Architecture:
  - Overview of the Cloud Architecture
  - Overview of the IoT Reference Apps
  - Overview of the Development Tools
  - Overview of SDK structure
- IoT Device Reference Application:
  - Overview of Application Architecture
  - Overview of Application Logic
  - Code Walk Through
- Action Items/Homework

SDK and Reference Architecture

Reference SDK

The Workshop will leverage a SDK (Software Development Kit) as a learning tool:

★ The SDK will be used as guide and a reference for the workshop materials.

★ Everything is documented in the SDK using README Markdown files.

★ There is JavaDoc (with UML diagrams) and PhpDoc that can be referenced as needed.

★ The SDK also provides you with all the scaffolding to get your Cloud development started:

- ✓ Spring Frame application template (can be imported into Eclipse).
- ✓ Laravel Application application template (can be imported into Eclipse).
- ✓ Step by step instructions for how to get started.

★ A fully functioning end to end system using one IoT Device application and two Enterprise applications are available (in code and operational). These are referred to as the IoT Reference Applications.

★ The Reference Applications were written in Python, Java, and PHP using languages you already know and can be used as is or used as example code.

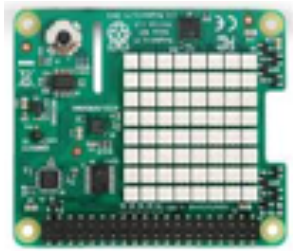
IoT Reference Applications

The following applications will be built in the Workshop using the Cloud based Development and Deployment Platform:

- ★ IoT Services App - Using Apache Tomcat or Redhat Wildfly Application Server, written in Java and leveraging the JAX-RS API, JSON, and the Spring Framework
- ★ IoT Reporting App - Responsive browser based app. written in PHP leveraging Bootrap (for responsive support), Guzzle (for HTTP Client to REST API), LavaCharts (for charting), and the Laravel Framework
  - ✳ An IoT Device Reference App. using a Raspberry Pi 3 and Sense HAT, written in Python, will be available for you to use in the Workshop.
  - ✳ If you build your own IoT Device App. using your own Raspberry Pi then I am assuming you will build this outside of the Workshop.

IoT Reference Architecture

IoT Device Application
(Sense HAT)



IoT Device Application
(Raspberry Pi)

REST API

HTTPS
HTTP Basic Auth.
Username/Password Auth.

IoT Services Application

REST API

HTTPS
HTTP Basic Auth.
Username/Password Auth.

IoT Reporting Application

SDK

- Developer Templ.
- Development Docs
- Tools

IoT Reference Applications

How do I get the code for the Reference Applications?

- **IoT Services App:**
  - Go to https://github.com/markreha/cloudservices.
    - Clone or download the repository from here.
    - See the README.md Markdown file for an overview.
- **IoT Reporting App:**
  - Go to https://github.com/markreha/cloudapp.
    - Clone or download the repository from here.
    - See the README.md Markdown file for an overview.
- **IoT Device App:**
  - Go to https://github.com/markreha/cloudpi.
    - Clone or download the repository from here.
    - See the README.md Markdown file for an overview.

SDK

- The SDK contains all the documentation, tools, etc. to get started building our Cloud based IoT applications.
- Go to https://github.com/markreha/cloudworkshop.
  - Clone or download the repository from here.
  - See the README.md markdown file for how to get started.

Folders of interest:

sdk	Root - See README.md
├ developer	App Templates, etc.
├ docs	All docs, see README.md
└ tools	Misc. doc tools gen. tools

Let's Walk Through the SDK!

Development Tools

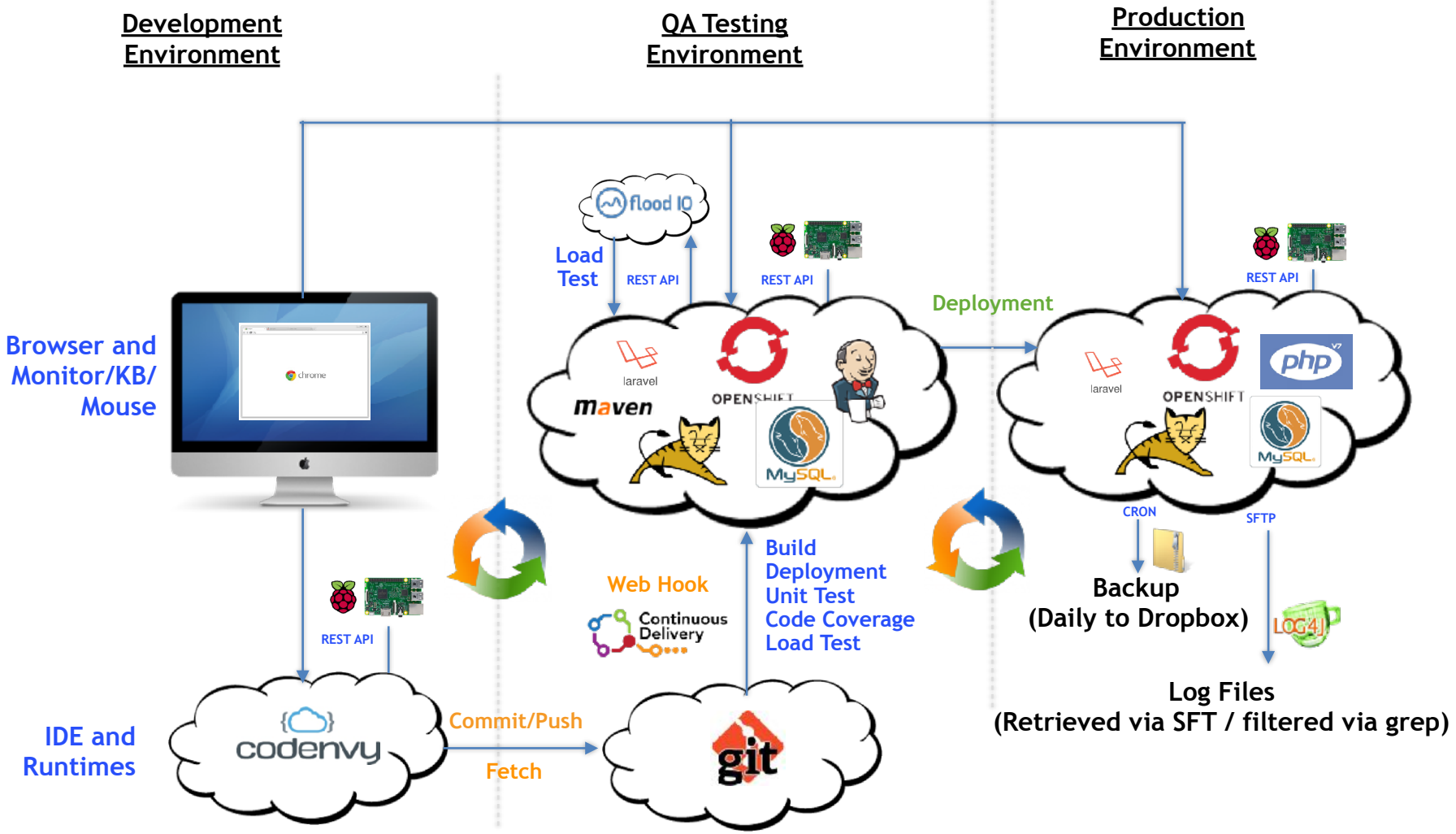
- For the Desktop you will need to install and setup a few tools……..
  - ✓ Eclipse Neon for EE and Tomcat or Wildfly.
  - ✓ Eclipse Neon for PHP.
  - ✓ MAMP Stack.
- For the Cloud you won't need to install anything……..
  - ✓ I would encourage you to develop and build some or all of your code using the Cloud based Codenvy IDE.
  - ✓ The beauty of using Codenvy is that you don't need to install anything! And you can quickly "rebuild" your environment if it gets messed up!
- You will also need to use a GIT client, such as SourceTree, or you can use a browser or you can use the GIT client built into Eclipse and Codenvy.

Deployment Environment

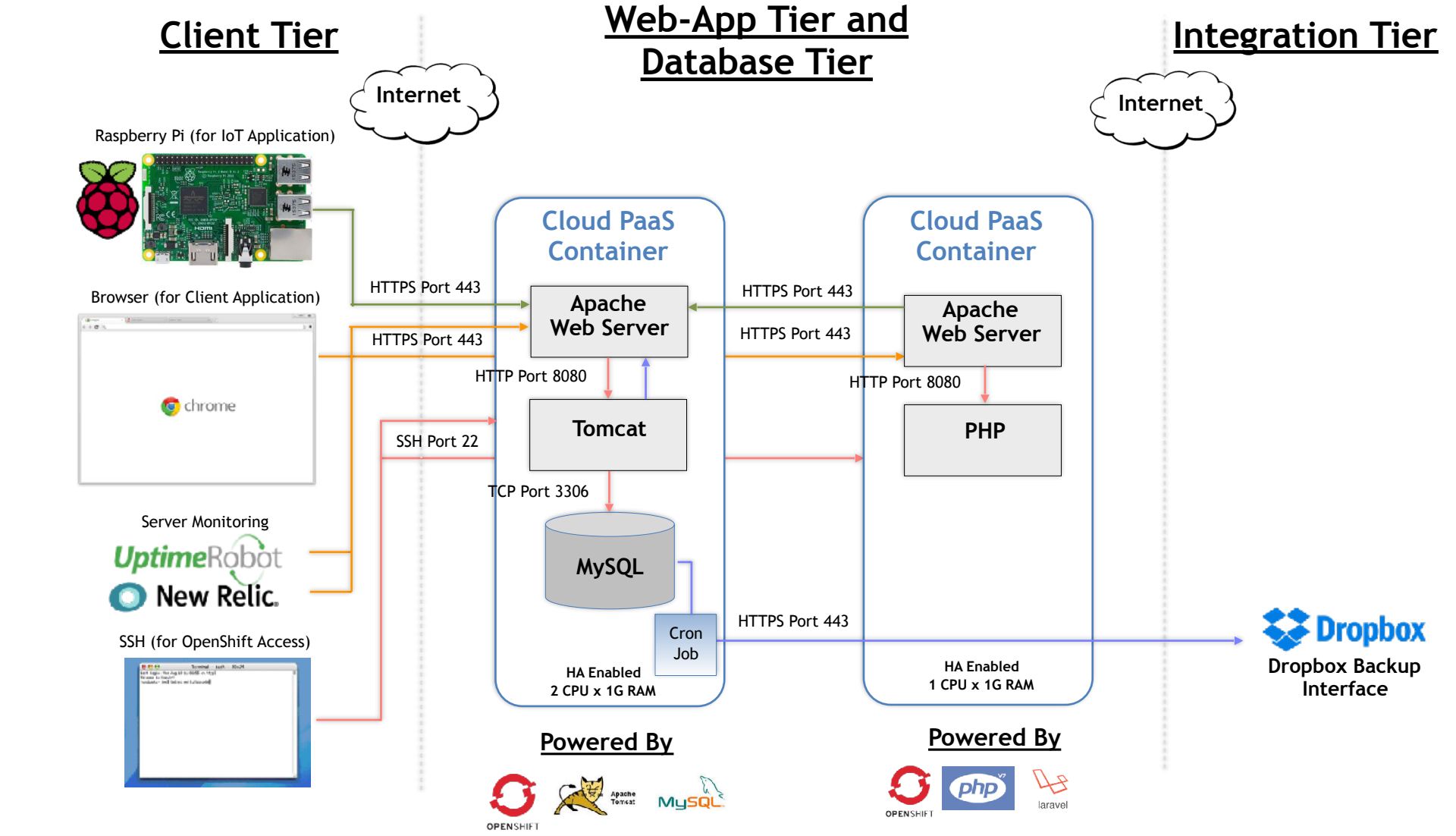
- You (or IT) won't need to install anything……..
  - ✓ We will use a Cloud PaaS.
  - ✓ We will use the public and free Redhat OpenShift Cloud and/or Microsoft Azure Cloud.
  - ✓ Once you have "procured" your Cloud Containers, setup your database, and completed some minimal configuration you will be ready to build and deploy your applications in the Cloud.
  - ✓ OpenShift uses Github as a source for all its builds.
    - ➡ Uses Maven if you are building a Java app.
    - ➡ Uses Composer if you are building a PHP app.
  - ✓ The beauty of the PaaS Cloud is that you (or IT) won't need to install anything to support application development!

OPENSHIFT by Red Hat Microsoft Azure GitHub Maven

Complete Cloud Based Platform



PaaS Cloud Based Architecture



IoT Device Reference Application

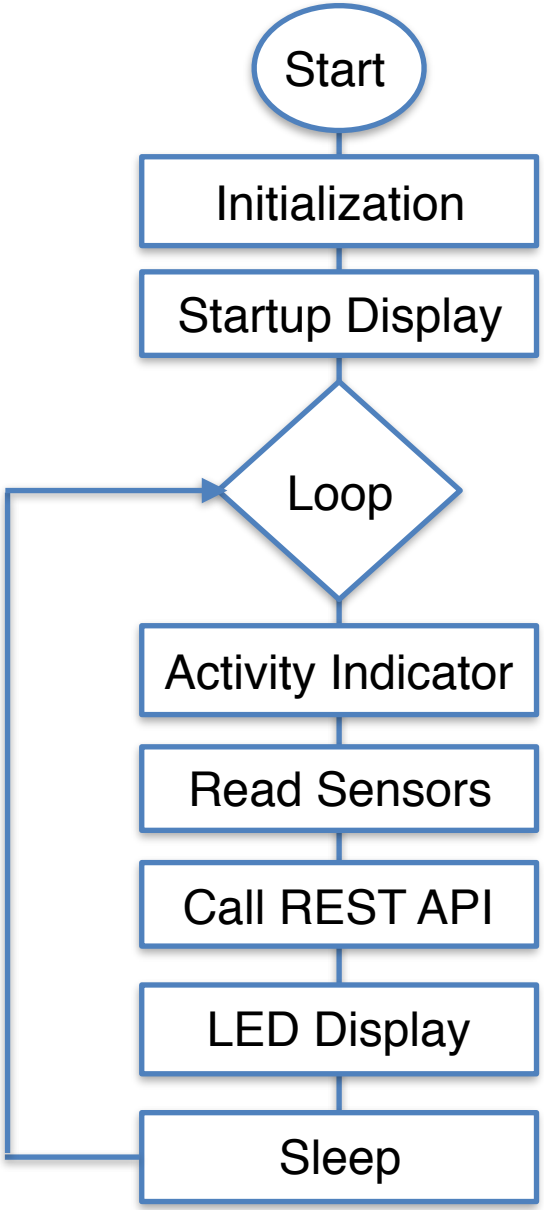
IoT Device

- IoT Device is a Rasberry Pi 3 with a Sense HAT.
- IoT Sense HAT features:
  - Temperature, Humidity, and Barometric pressure
  - Gyroscope, Accelerometer, and Magnetometer
  - LED Display
  - Joystick

IoT Device Reference Application

- IoT Device Reference Application is written in Python 3.
  - Polls for Temperature, Humidity, and Barometric pressure IoT Data every 30 minutes
  - Then calls the IoT Services REST API's to save the data.
- IoT Device Reference Application is actually pretty simple because all of the Sense HAT API's are built into the Raspberry Pi OS (Raspbian Linux).
- I did most of the development by connecting to the Raspberry Pi using a Display via HDMI cable and keyboard and mouse via USB.
- However, now I just connect to the Raspberry Pi remotely using VNC.

IoT Device Application Logic



Initialize logging, read config file, init local variables.

Displays G, C, U on Sense HAT LED Display.
Display App Startup Message on the Console.

Loop Forever.

Display Activity Dot (.) on the Console.

Read IoT Data from Sense HAT.
Round data and do conversions.

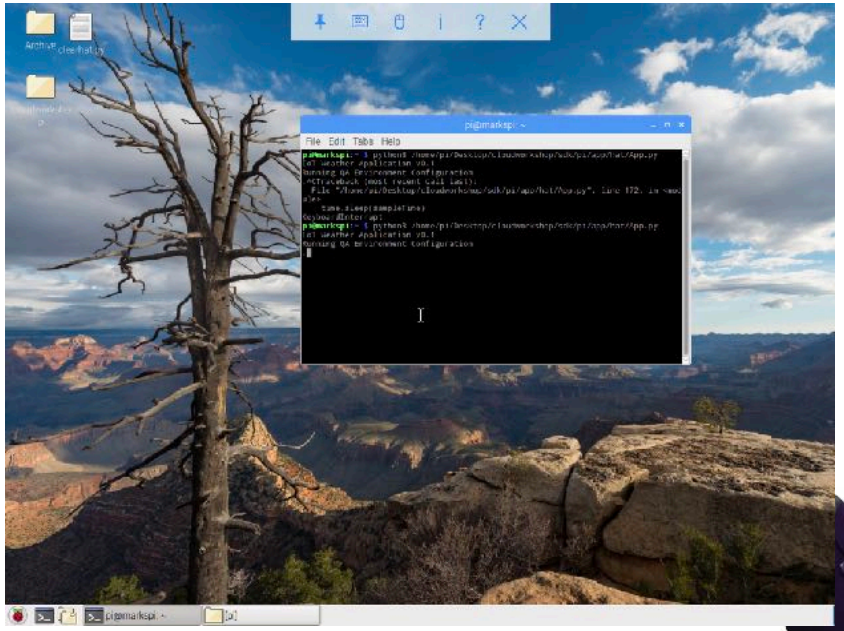
POST data to IoT Services App.

Display Purple pixel (no error) or Red pixel (error) on Sense HAT LED Display.

Sleep for specified time.

IoT Device Application Code Walkthrough

Sense HAT Raspberry Pi 3



Connected via VNC

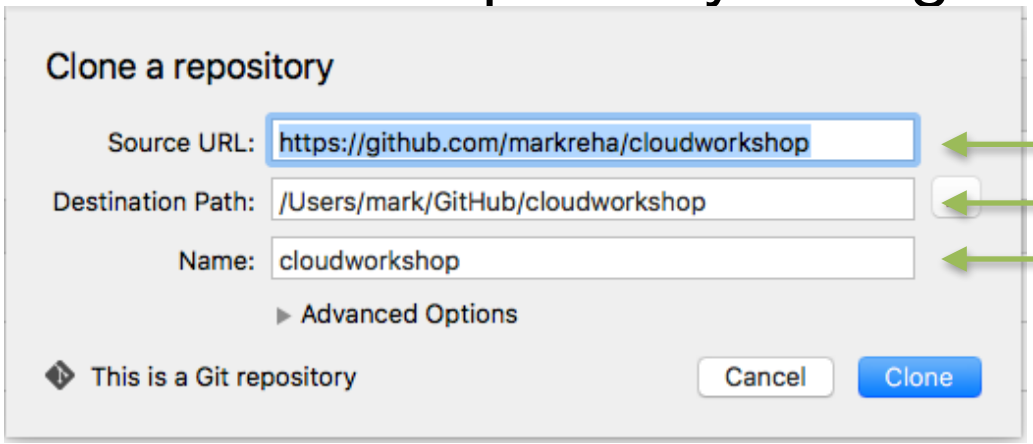
Let's Walk Through the Code!

Action Items / Next Steps

How Do I Clone the Workshop Repositories?

1. Click the Remote Tab -> +New -> Clone from URL.

2. Fill in the Clone Repository dialog as follows.



Source Repo URL

Local Repo Path

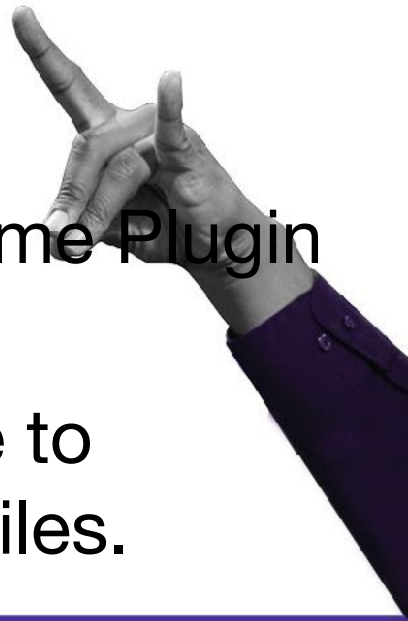
Local Repo Name

3. Click the Clone button.

4. Click the Local Tab. Commit and Push to YOUR Repo.

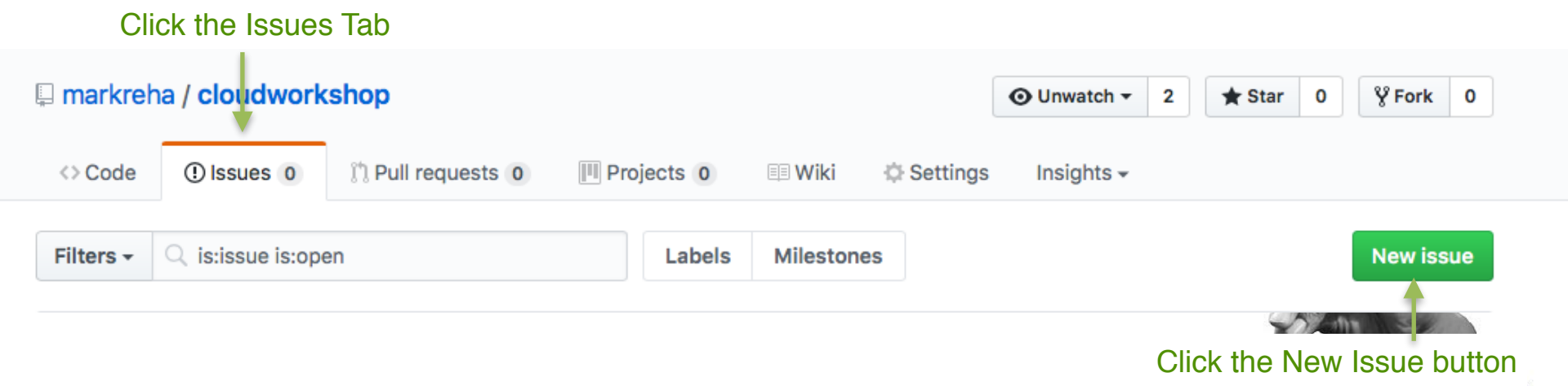
How Can I View Local Markdown Files?

1. All of the README files in Github are Markdown files.
2. They are rendered by default if you go to the web version of Github.
3. If you clone the SDK to your local file system you can still view the README Markdown files.
   - You must use the Chrome Browser.
   - Then install the Markdown Preview Chrome Plugin from the Chrome Web Store.
   - Use the File->Open File menu in Chrome to navigate and view your Markdown (md) files.



How Do I Report a Bug?

1. Go the appropriate Github repository that has a bug.
2. Report an Issue as follows:



3. Fill out the New Issue form and then click the Submit new issue button.

Action Items / Homework

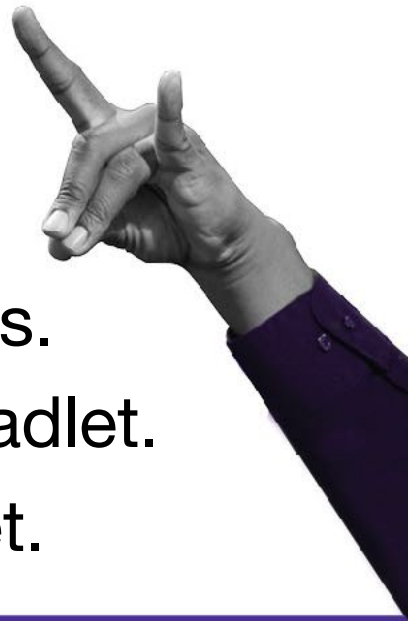
1. Setup the following Cloud accounts:
    - Github account (go to https://github.com).
    - Codenvy account (go to https://codenvy.com).
    - OpenShift account (go to https://www.openshift.com).
        - Be prepared to setup 2 accounts so you can deploy Java/MySQL and PHP apps.
2. Install the following tools if you plan on using the desktop for development (optional if using Codenvy IDE):
    - Eclipse EE edition and Tomcat 8.5.
    - Eclipse PHP edition.
    - MAMP and MySQL Workbench.
    - Postman.
3. Start using the SDK:
    - Clone the SDK from the 'cloudworkshop' Github repository (see Clone Repo slide).
    - Review the README.md in the *root* directory of the SDK or the 'cloudworkshop' Github repository.
    - Review the README.md in the *docs/development* directory in the SDK or the Github repository.
4. Get ready for Workshop Session #2:
    - Go thru Maven Tutorials (see Reference Resources slide).
    - Go thru Spring Tutorials (see Reference Resources slide).
    - Go thru REST Tutorials. (see Reference Resources slide)
    - Clone the IoT Services Reference Application from the 'cloudservices' Github repository.
    - Review the IoT Services Reference App. code.

Reference Materials

- Maven Tutorials:
  - ✓ Tutorial from TutorialsPoint
  - ✓ What Is Maven?
  - ✓ Maven In 5 Minutes
  - ✓ Maven Home Page
- Spring Tutorials:
  - ✓ Tutorial on Spring Core from TutorialsPoint
  - ✓ Tutorial on Spring JDBC from TutorialsPoint
  - ✓ Spring Framework Documentation
  - ✓ Spring Framework Home Page
- REST Tutorials:
  - ✓ Tutorial from TutorialsPoint
  - ✓ Tutorial from Oracle
  - ✓ JAX-RS Specification

Next Up

- Workshop Session 2 will be scheduled for next week.
- Agenda:
  - Q/A.
  - Review the IoT Services App. Architecture.
  - Overview of Spring, JAX-RS, and Maven.
  - Review the IoT Services App. Code.



- We will be monitoring our Padlet for questions.
  - Post your tool/account questions in the Padlet.
  - Post your technical questions in the Padlet.

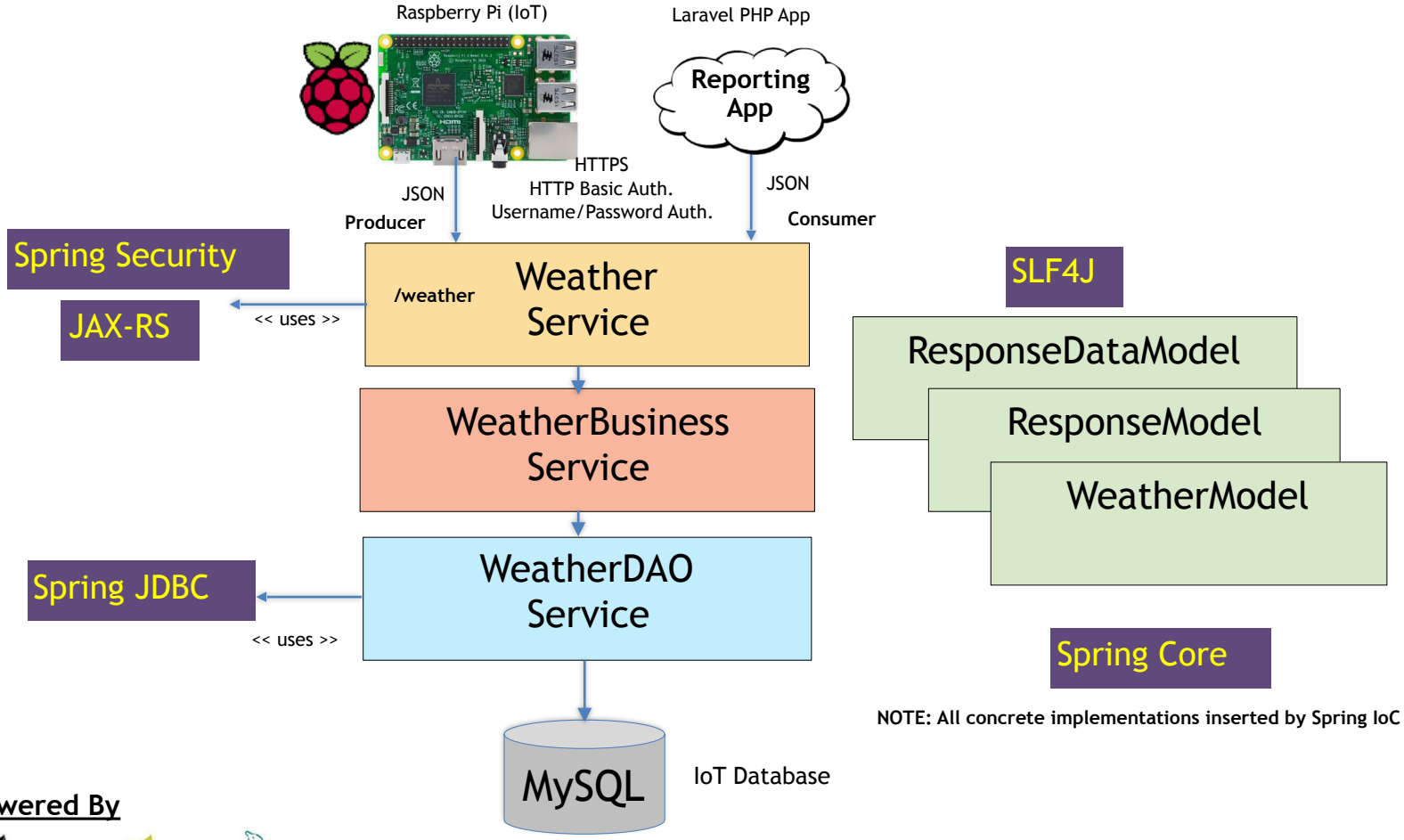
Workshop Session 2

Workshop Session 2

- IoT Services Reference Application:
  - Overview of the Application Architecture
  - Overview of the Spring Framework
  - Overview of JAX-RS
  - Code Walk Through
  - Maven Walk Through
- Action Items/Homework

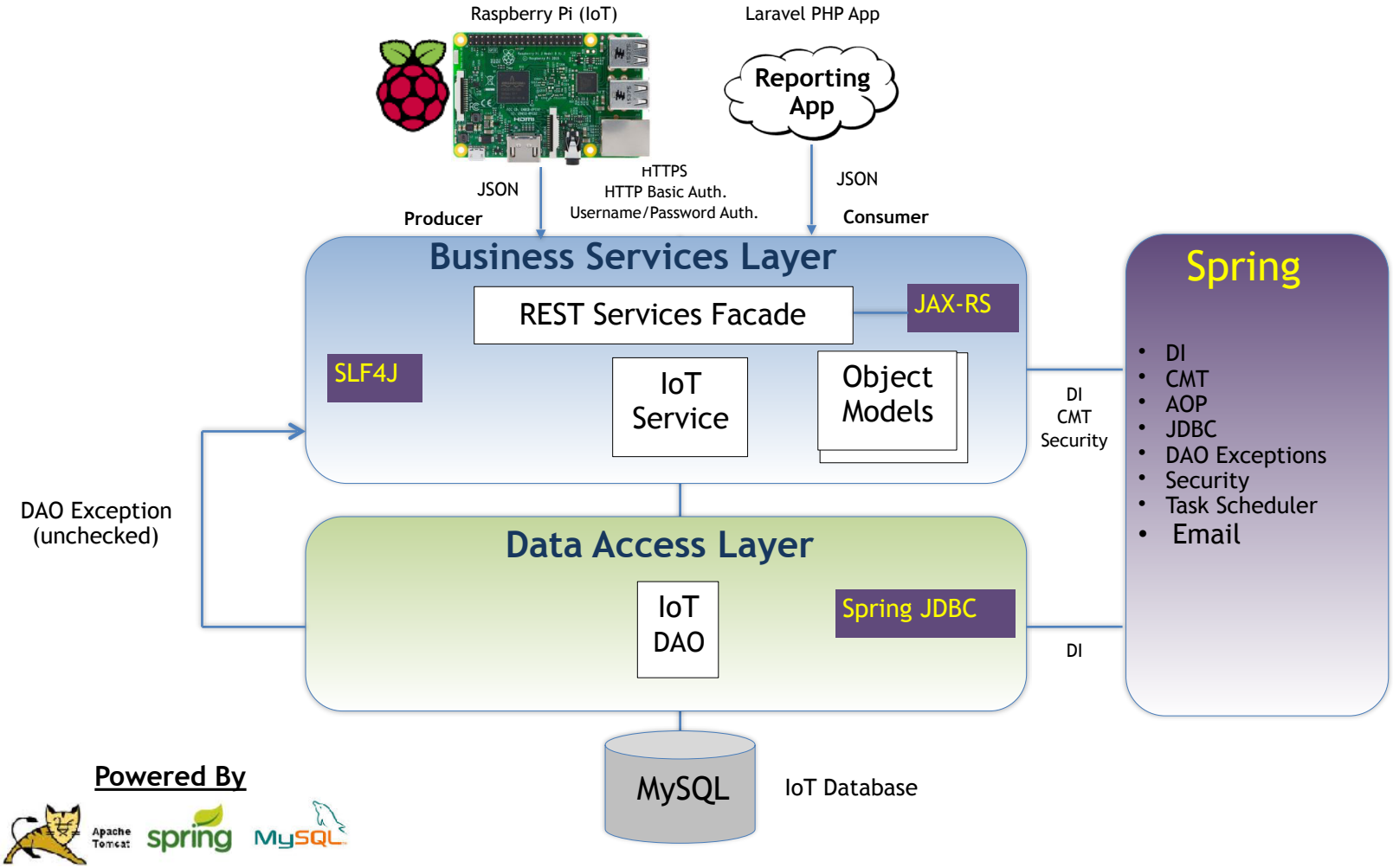
Application Architecture

IoT Services App Logical Architecture



Powered By

IoT Services App Logical Architecture



Powered By

IoT Weather Service REST API's

1. /test: Testing Method

   HTTP Method: GET

   Consumes: Nothing

   Produces: JSON of ResponseModel

2. /save: Saves IoT Weather Data

   HTTP Method: POST

   Consumes: JSON of WeatherSensorModel

   Produces: JSON of ResponseModel

3. /get/{device}/{from}/{to}: Returns IoT Weather Data

   HTTP Method: GET

   Consumes: URI parameters for Device ID, From Date, and To Date

   Produces: JSON of ResponseDataModel

See JavaDoc and Swagger API in the SDK!

Code Walk Through

What Is The Spring Framework?

- Spring is an alternative Java stack for building web and enterprise applications.
- Spring has essentially replaced Enterprise Java in the industry.
- Spring consists of a number of “modules”. We will use Spring Core, Spring Security, and Spring JDBC.
- Spring supports a “light weight” POJO based programming model using annotations and/or XML configuration.
- All POJO classes are referred to and defined as SpringBeans.
- SpringBeans can configured and injected using IoC.
- Spring apps can be deployed on Apache Tomcat and do not require a full J2EE compliant app. server.

What Is JAX-RS?

- JAX-RS is a specification for building REST based services in Java.
- An implementation of the specification is included in all Enterprise Java stacks.
- There are a number of other implementations that are available if you are not using a full Enterprise Java stack (for example Spring).
- We are going to use Jersey (and Jackson) for our implementation.
- JAX-RS makes building REST based services easy by simply using a few annotations in your code.

JAX-RS Code Example

URI to Service Class

Simply a POJO class

HTTP GET (or @POST)

URI to Service Method

MIME Type for return type (also see @Consumes)

Jackson converts to JSON for you

```
@Path("weather")
public class RestService
{
    @GET
    @Path("/test")
    @Produces("application/json")
    public ResponseModel test()
    {
        // Return a Test Response
        ResponseModel response = new ResponseMod
        return response;
    }
}
```

Create a POJO and markup your code using JAX-RS Annotations.
Import annotations from javax.ws.rs.* package.
You will also have to do some one time configuration.

Jersey JAX-RS Configuration

```
<!-- Jersey Servlet -->
<servlet>
    <servlet-name>Jersey Web Services</servlet-name>
     <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
     <init-param>
        <param-name>jersey.config.server.provider.packages</param-name>
        <param-value>edu.gcu.web.service</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>Jersey Web Services</servlet-name>
    <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Any URI with /rest/ will get handled by the Jersey Servlet

This configuration goes in your web.xml file!

REST Service URI

/cloudservices/rest/weather/test

HTTP GET

App Name

Request Handled by Jersey Servlet

RestService Class Instantiated

No Parameters!

RestService.test() Invoked

Jackson converts all service method parameters and return values to JSON for you.

What is Spring Security?

Spring Security is a powerful framework to secure Spring applications. We will configure Spring to force HTTP Basic Authentication scheme for all of our REST API's.

```
<!-- Security for REST Services -->
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/rest/*</url-pattern>
</filter-mapping>
```

Any request URI with /rest/ will get handled by the Spring Proxy Filter

This configuration goes in your web.xml file!

Spring Security Configuration

```
<sec:authentication-manager alias="casAuthenticationManager" >
    <sec:authentication-provider>
        <sec:user-service>
         <sec:user name="CloudWorkshop" password="dGVzdHRlc3Q=" authorities="ROLE_SYSTEM" />
     </sec:user-service>
</sec:authentication-provider>
</sec:authentication-manager>
<sec:http auto-config="true">
    <sec:intercept-url pattern="/rest/**" access="ROLE_SYSTEM" />
    <sec:http-basic />
</sec:http>
```

Use HTTP Basic Authentication

Any URI with /rest/ will require HTTP Basic Authentication

Security Credentials

This configuration goes in your securityContext.xml file!

How Does Spring Support IoC?

Spring Core contains an IoC Container and is really the only way to develop in Spring (leveraging SpringBeans).

SpringBean Definition

Inject Business Service SpringBean into this SpringBean

```
<bean id="weatherService" class="edu.gcu.business.WeatherService">
    <property name="dao">
        <ref bean="weatherDao" />
    </property>
</bean>
<bean id="restServicesBean" class="edu.gcu.web.service.RestService">
    <property name="service">
        <ref bean="weatherService" />
    </property>
</bean>
```

This configuration goes in your applicationContext.xml file!

Spring IoC Code Example

Variable to inject (note an interface)

Use setter injection

```
public class RestService
{
    static WeatherServiceInterface service;

    // ***** Dependencies and Helper Functions *****
    public void setService(WeatherServiceInterface service)
    {
        RestService.service = service;
    }
}
```

You can also use Annotations versus using XML!

What Is Spring JDBC?

Spring JDBC is a simple framework to execute SQL statements using Spring JDBC API's.

Action	Spring	You
Define connection parameters.		X
Open the connection.	X	
Specify the SQL statement.		X
Declare parameters and provide parameter values		X
Prepare and execute the statement.	X	
Set up the loop to iterate through the results (if any).	X	
Do the work for each iteration.		X
Process any exception.	X	
Handle transactions.	X	
Close the connection, statement and resultset.	X	

No coding required to handle connections, cleanup, etc.

Once you do a bit of configuration you can then focus on writing your database CRUD code!!

Spring JDBC Configuration

SpringBean

Use Apache DB Connection Pool

Use MySQL DB JDBC Driver

```
<bean id="dataSource" destroy-method="close" class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://${db.hostname}/iot" />
    <property name="username" value="${db.username}" />
    <property name="password" value="${db.password}" />
</bean>
```

SpringBean

Inject Datasource SpringBean into this SpringBean

```
<bean id="weatherDao" class="edu.gcu.dao.WeatherDAO">
    <property name="dataSource">
        <ref bean="dataSource" />
    </property>
</bean>
```

This configuration goes in your applicationContext.xml file!

Spring JDBC Code Example

```
@SuppressWarnings("unused")
public class WeatherDAO implements WeatherDAOInterface
{
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplateObject;

    public void setDataSource(DataSource dataSource)
    {
        this.dataSource = dataSource;
        this.jdbcTemplateObject = new JdbcTemplate(dataSource);
    }
}
```

Variable to inject

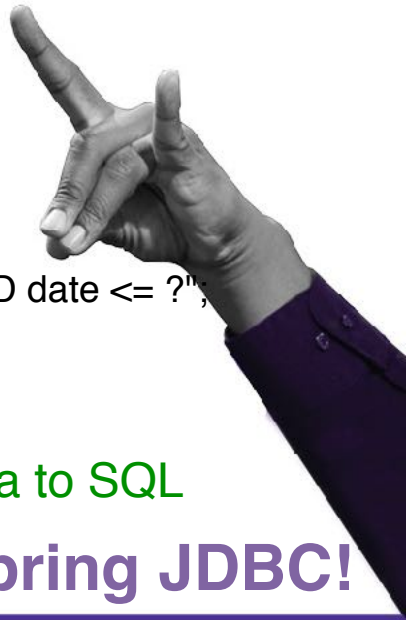
Use setter injection

Create JDBC Template using this Datasource

Just like Prepared Statements

```
String sql = "SELECT * FROM weather WHERE device_id=? AND date >= ? AND date <= ?";
SqlRowSet srs = jdbcTemplateObject.queryForRowSet(sql, deviceID, from, to);
while(srs.next())
{}
```

Use queryForRowSet() API for SQL queries and to bind your data to SQL



See update() API for doing a SQL insert using Spring JDBC!

A Few Misc. Things

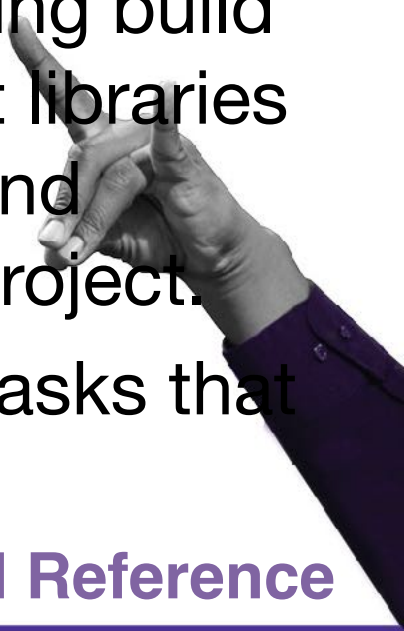
- Handling time and dates in Java needs to be done properly due to timezones
- Remember the time zone by default and unless you configure or code to a specific timezone will be set by the JVM and obtained from the OS.
- If you deploy your application to a Cloud Server that is running in Europe or EST then your time and dates may need to be adjusted for a desired timezone to meet your requirements.
- I had to do this for the IoT Reporting Application!

Let's look at the code to handle PST timezone.

Maven Overview

What Is Maven?

- Maven is a build tool. Maven is a dependency management system.
- You configure your application using an XML file (i.e. pom.xml) that goes in the root of your project.
- The pom.xml file defines plugins used during build process, what files to build/package, what libraries and their versions that you want to build and package with, and how to package your project.
- Maven is driven by goals that define that tasks that you want to perform during a build.



Let's look at the Maven file for the Template and Reference App!

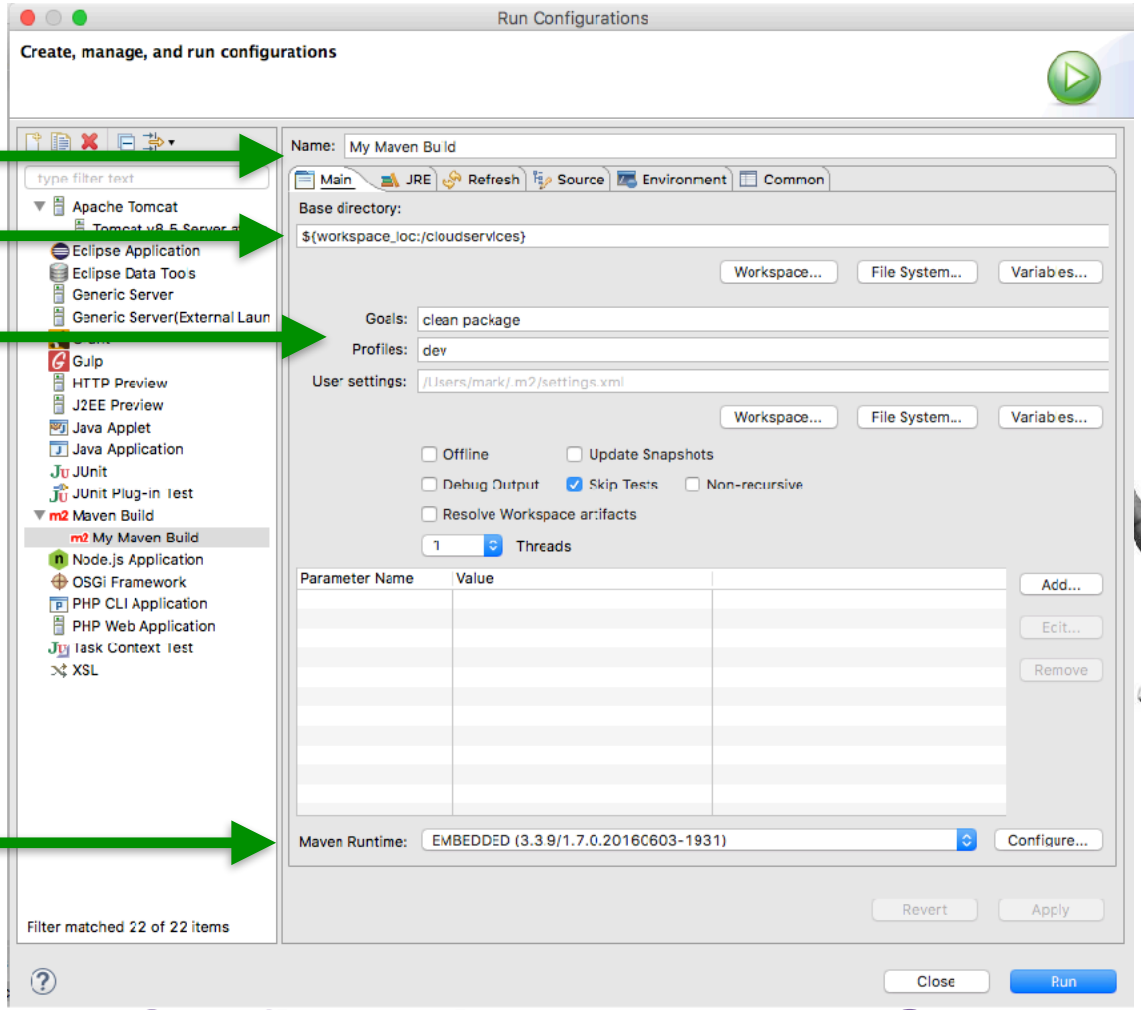
How Do I Do a Maven Build in Eclipse?

Run Name

Workspace Name

Maven Goals and Profile

Use Maven built into Eclipse



Under Run->Run Configurations…..menu. Select Maven Build.

How Do I Do a Maven Build in Codenvy?

```
mvn clean package -f ${current.project.path} -Pdev
```

Maven CLI

Maven Goals

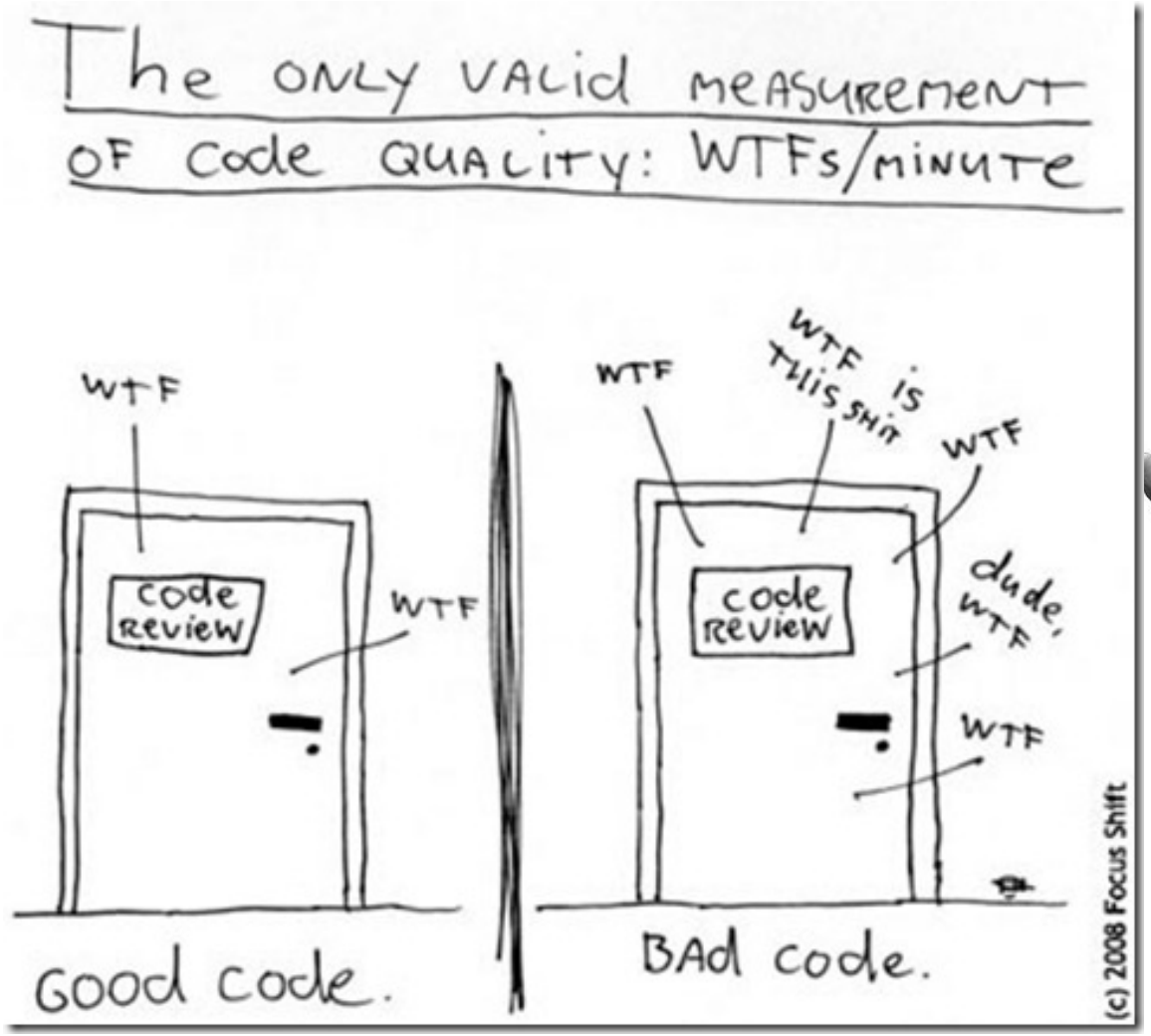
Working Directory

Maven Profile

OpenShift
also uses
Maven for its
builds!!

Setup a Custom Command and invoke via CLI.

Reference App. Code Walk Through



Action Items / Next Steps

Action Items / Homework

1. Use the SDK and Reference Apps:
    - Clone the IoT Service Reference app or the Template app.
    - Push the code to your Github repository.
    - Review the code.
    - Review the README.md in the ‘cloudservices’ Github repository.
    - Review the README.md in the *docs/development* directory from the SDK.
    - Import your app into Eclipse and/or the Codenvy IDE.
2. Get ready for Workshop Session #3:
    - Go thru JAX-RS Tutorials (see Reference Resources slide).
    - Go thru Codenvy Tutorials (see Reference Resources slide).
    - Go thru OpenShift Tutorials. (see Reference Resources slide).

Reference Materials

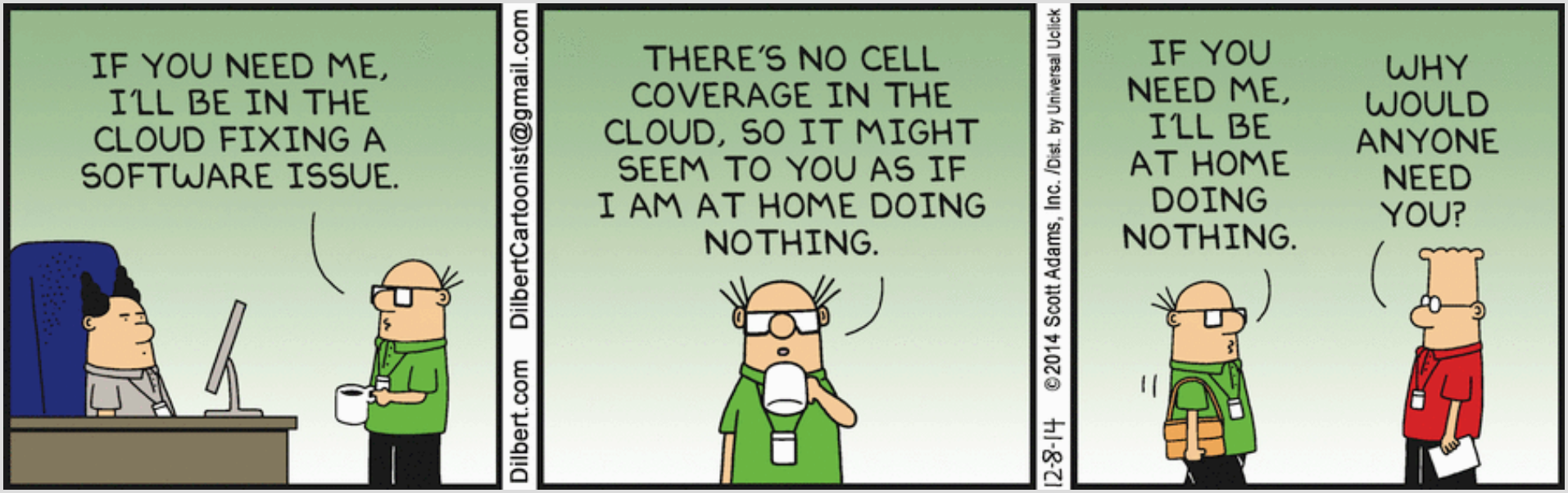
- Codenvy Tutorials:
  - ✓ Tutorial from Codenvy
- OpenShift Tutorials:
  - ✓ Docs from Redhat
- JAX-RS Tutorials:
  - ✓ Jersey + Jackson Tutorial
  - ✓ JAX-RS Using Jersey Tutorial
- Spring Documentation:
  - ✓ Spring JDBC Docs

Next Up

- Workshop Session 3 will be scheduled for this coming Thursday.
- Agenda:
  - Q/A.
  - Build the IoT Services App. in Eclipse and Codenvy.
  - Test the IoT Service App. API's.
  - Deploy the IoT Services App. to OpenShift.

- We will be monitoring our Padlet for questions.
  - Post your tool/account questions in the Padlet.
  - Post your technical questions in the Padlet.

Workshop Session 3



Workshop Session 3

- IoT Services Reference Application:
  - Development Tooling Setup and Build
  - Testing and Documenting your API's
  - Cloud Deployment (time permitting)
- Action Items/Homework

Build The IoT Services App.

How Do I Build In Eclipse?

- See the README.md file in the *docs/development* directory of the SDK!

Let's import the Reference App!

How Do I Build In Codenvy?

- See the README.md file in the *docs/development* directory of the SDK!

Let's import the Reference App!

How Do I Build & Deploy in OpenShift?

- See the README.md file in the *docs/development* directory of the SDK!

Let's build and deploy the Reference App!

REST API Tools

What Tools Do I Need For Developing REST API's?

- Testing:
  - Use Postman!
  - You can use this tool during your development to simply test out your API's (either GET or POST).
  - You can also use this tool to setup Unit Tests as a Test Suite to test one or more of your API's.
- Documentation:
  - Use Swagger!
  - You should always document your API's in JavaDoc and/or Swagger.
  - See the example in the SDK that I built and published.

Hands On in the Cloud

Action Items / Next Steps

Action Items / Homework

1. Use the SDK and Reference Apps:
   - Get your app in Eclipse and/or the Cloudenvy IDE to build and deploy on Tomcat.
   - Test your code using Postman.
2. Get ready for Workshop Session #4:
   - Go thru PHP Tutorials (see Reference Resources slide).
   - Clone the IoT Reporting Reference Application from the ‘cloudapp’ Github repository.
   - Review the IoT Reporting Reference App. code from the ‘cloudapp’ Github repository.

Reference Materials

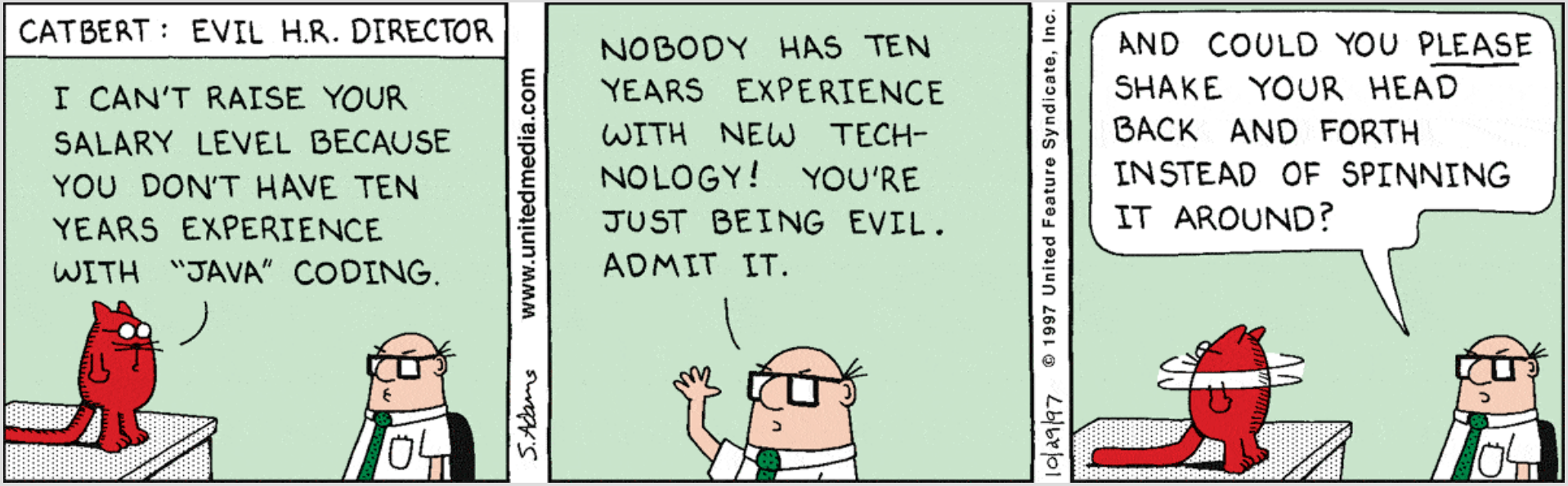
- Laravel PHP Tutorials:
  - ✓ Guzzle Library Docs
  - ✓ Lavacharts Library Docs
- Composer Tutorials:
  - ✓ Online Tutorial 1
  - ✓ Online Tutorial 2

Next Up

- Workshop Session 4 will be scheduled for next week.
- Agenda:
  - Q/A.
  - Review the IoT Reporting App. Architecture.
  - Overview of Guzzle, Lavacharts, and Composer.
  - Review the IoT Reporting App. code.

- We will be monitoring our Padlet for questions.
  - Post your tool/account questions in the Padlet.
  - Post your technical questions in the Padlet.

Workshop Session 4

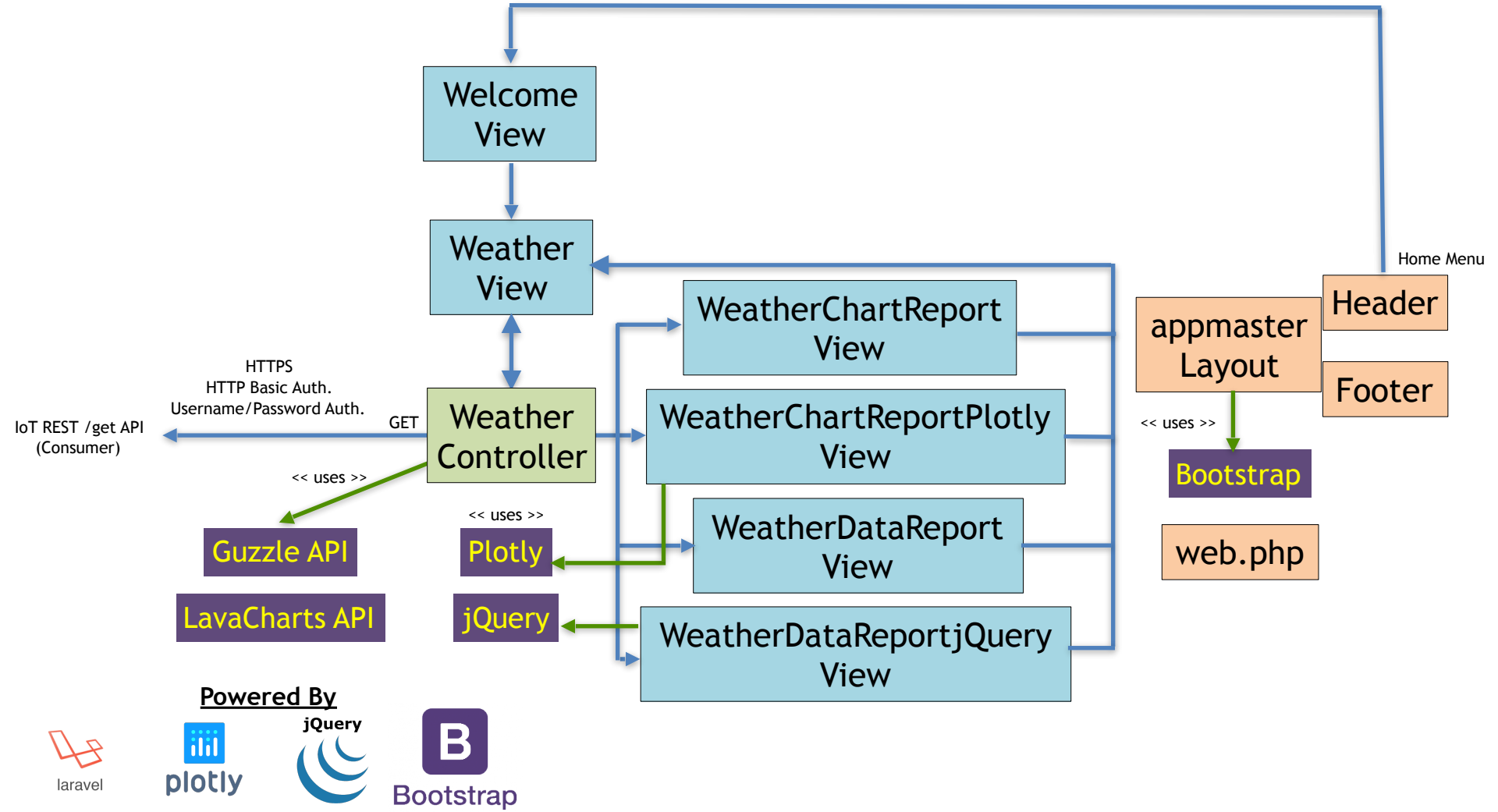


Workshop Session 4

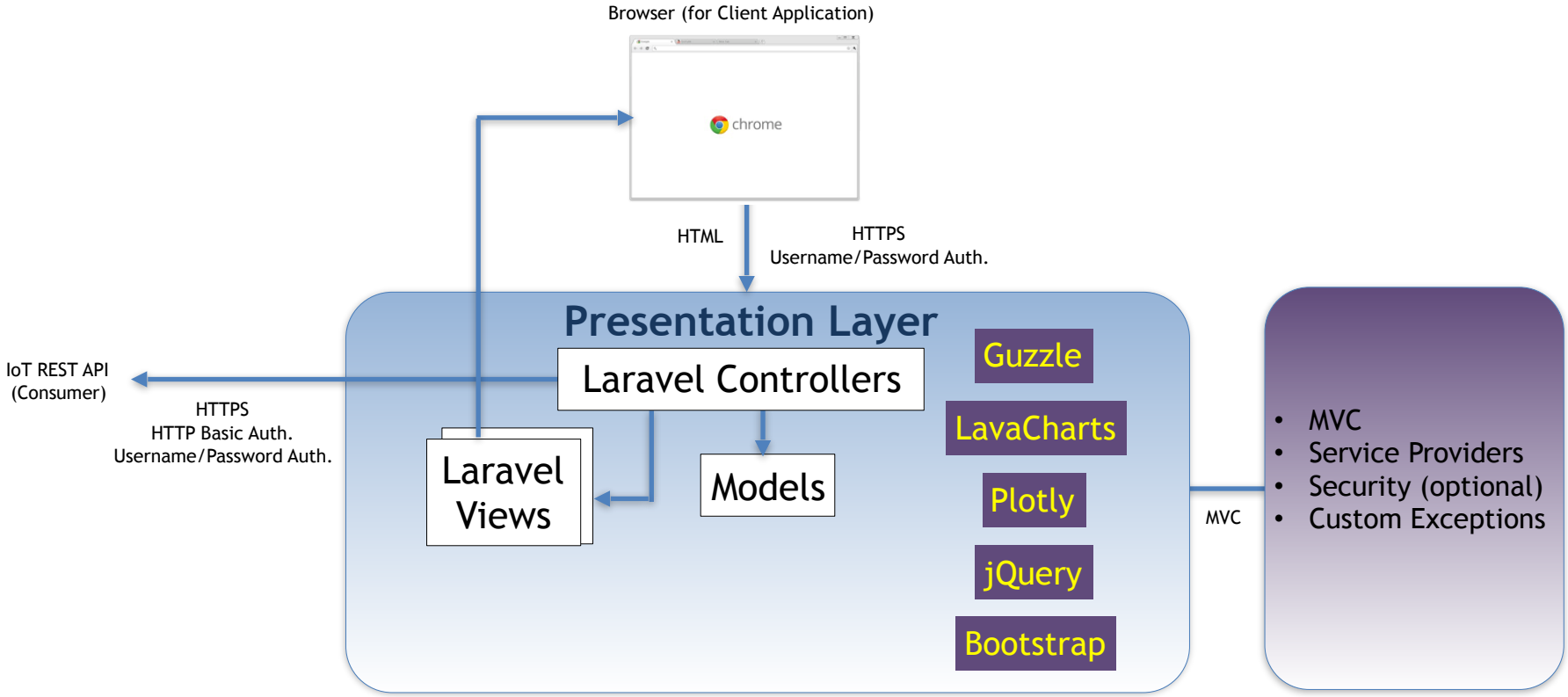
- IoT Reporting Reference Application:
  - Overview of the Application Architecture
  - Overview of Guzzle
  - Overview of Lavacharts
  - Code Walk Through
  - Composer Walk Through
- Action Items/Homework

Application Architecture

IoT Reporting App Logical Architecture



IoT Reporting App Logical Architecture



Powered By

php v7 laravel plotly jQuery Bootstrap

Code Walk Through

What Is Guzzle?

- Guzzle is PHP library that implements a HTTP Client.
- Guzzle makes it easy to send HTTP requests for making REST based API calls.
- Supports POST and GET.
- Supports HTTP Basic Authentication.
- Supports setting HTTP Parameters.
- Supports handling HTTP Responses.
- Is easy to use.

What Is Lavacharts?

- Lavacharts is PHP library that implements charting functionality.
- Lavacharts is a wrapper for Google's powerful Javascript Chart API.
- The Lavacharts library wraps the work of generating the proper Javascript so you can focus on the data and not the Javascript in page.
- Supports Data Tables, Line Charts, Bar Charts, Pie Charts, and more.
- Is pretty easy to use. Also see Plotly!!

Guzzle Code Example

Guzzle HTTP Client

Build REST API URI

Invoke REST API

Check HTTP Response Code

Convert JSON to Object

```
// Call Web API to get Weather Data
$serviceURL = "http://cloudservices-workshop.1d35.starter-us-east-1.openshifta
$client = new Client(['base_uri' => $serviceURL]);

$api = "get";
$device = 0;
$uri = $api . "/" . $device . "/" . $from . "/" . $to;

$username = "CloudWorkshop";
$password = “dGVzdHRlc3Q=";
$response = $client->request('GET', $uri, ['auth' => [$username, $password]]);

// Process API Response
if($response->getStatusCode() == 200)
{
    // Convert JSON to Objects and render report
    $jsonObj = json_decode($json = $response->getBody(), true);
```

This code can go in one of your Laravel Controllers.
Import Guzzle Library from GuzzleHttp\Client namespace.
You will also have to configure Composer.

Lavacharts Code Example

Convert JSON to Object

```
$jsonObj = json_decode($json = $response->getBody(), true);
```

Lavacharts instance, DataTable instance, and Chart Columns

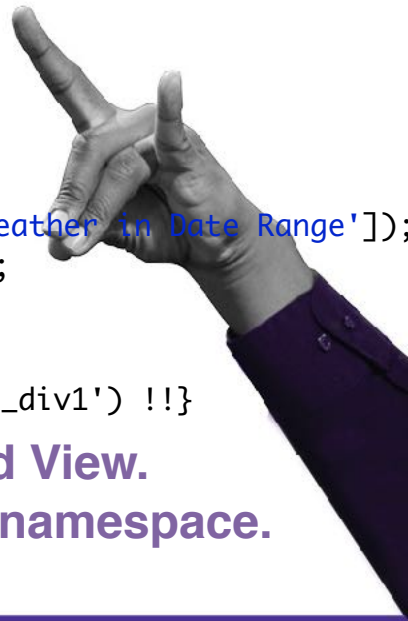
```
// Build a Lavachart Datatable, Line Chart, and pass the Chart to the View to
$lava = new Lavacharts;
$weatherData = $lava->DataTable();
$weatherData->addStringColumn('Date')
            ->addNumberColumn('Temp')
            ->addNumberColumn('Pressure')
            ->addNumberColumn('Humidity');
```

Add Chart Data to DataTable

```
foreach ($jsonObj['data'] as $item)
{
    $weatherData->addRow([$item['date'],
                          $item['temperature'],
                          $item['pressure'],
                          $item['humidity']]);
}
```

Create Line Chart and pass to View

```
$lava->LineChart('Temps', $weatherData, ['title' => 'Weather in Date Range']);
return view("weatherReportChart")->with("lava", $lava);
```



Render Chart View

```
{!! $lava->render('LineChart','Temps', 'weather_div1') !!}
```

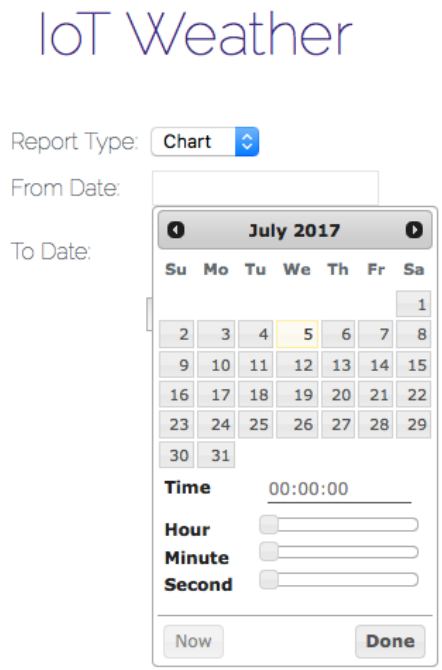
This code can go in one of your Laravel Controllers and View.
Import Lavacharts Library from Khill\Lavacharts\Lavacharts namespace.
You will also have to configure Composer.

jQuery Time-picker Code Example

Initialize each of the time-pickers

Bind each time-picker to each edit text control

```
<script>
      $('#fromDate').datetimepicker({
            dateFormat: 'yy-mm-dd',
            timeFormat: 'HH:mm:ss',
            separator: ' ',
            showTimezone: false
      });
      $('#toDate').datetimepicker({
            dateFormat: 'yy-mm-dd',
            timeFormat: 'HH:mm:ss',
            separator: ' ',
            showTimezone: false
      });
      $('#fromDate').datetimepicker();
      $('#toDate').datetimepicker();
</script>
```



This code can go in one of your Laravel Views.
Import the jQuery Timepicker JS files from the Reference App.
Remember your end user and the device they are using.
Make your apps easy to use!

Composer Overview

What Is Composer?

- Composer is a dependency manager for PHP.
- Composer will manage the dependencies you require on a project by project basis. This means that Composer will pull in all the required libraries, dependencies and manage them all in one place.
- You configure your application using a JSON file (i.e. composer.json) that goes in the root of your project.
- There are also plug-ins available for Eclipse.

How Do I Configure Dependencies in Composer?

composer.json

Add your dependencies and version to the “require” section

```
"require": {
        "php": ">=5.6.4",
        "guzzlehttp/guzzle": "~6.0",
        "khill/lavacharts": "^3.1",
        "laravel/framework": "5.4.*",
        "laravel/tinker": "~1.0"
    },
```

To install the components into your project run composer from the command line using the install option

```
php composer.phar install
```

OpenShift also uses Composer for its “builds”!!

Let’s look at the Composer file for the Reference App

Reference App. Code Walk Through

Action Items / Next Steps

Action Items / Homework

1. Use the SDK and Reference Apps:
    - Clone the IoT Reporting Reference app or the Template app.
    - Push the code to your Github repository.
    - Review the code.
    - Review the README.md in the ‘cloudapp’ Github repository.
    - Review the README.md in the *docs/development* directory from the SDK.
    - Import your app into Eclipse and/or the Cloudenvy IDE.
2. Get ready for Workshop Session #5:
    - Go thru Composer Tutorials (see Reference Resources slide).
    - Go thru Codenvy Tutorials (see Reference Resources slide).
    - Go thru OpenShift Tutorials. (see Reference Resources slide).

Reference Materials

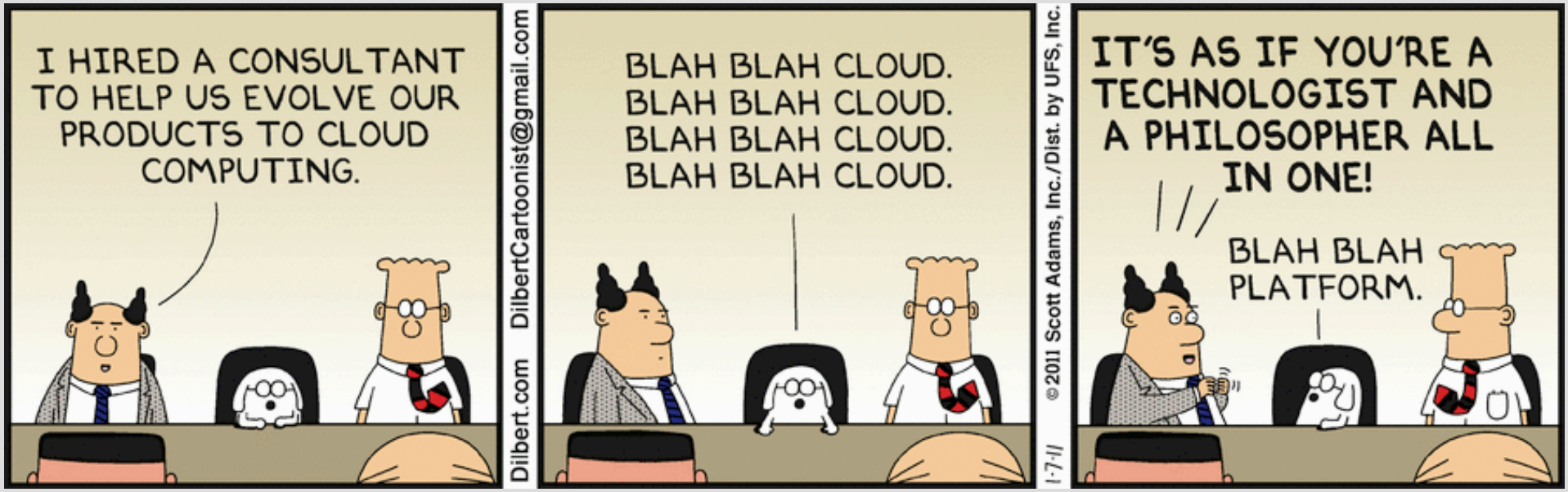
- Codenvy Tutorials:
  - ✓ Tutorial from Codenvy
- OpenShift Tutorials:
  - ✓ Docs from Redhat
- Composer Docs:
  - ✓ Composer Online Docs
- jQuery Timepicker Docs:
  - ✓ jQuery Timepicker Github Docs
  - ✓ jQuery Timepicker Docs

Next Up

- Workshop Session 5 will be scheduled for this coming Thursday.
- Agenda:
  - Q/A.
  - Build the IoT Reporting App. in Eclipse and Codenvy.
  - Test the IoT Reporting App. (end to end).

- We will be monitoring our Padlet for questions.
  - Post your tool/account questions in the Padlet.
  - Post your technical questions in the Padlet.

Workshop Session 5



Workshop Session 5

- IoT Reporting Reference Application:
  - Development Tooling Setup and Build
  - Cloud Deployment
- Action Items/Homework

Build The IoT Reporting App.

How Do I Build In Eclipse?

- See the README.md file in the *docs/development* directory of the SDK!

Let's import the Reference App!

How Do I Build In Codenvy?

- See the README.md file in the *docs/development* directory of the SDK!

Let's import the Reference App!

How Do I Build & Deploy in OpenShift?

- See the README.md file in the *docs/development* directory of the SDK!

Let's build and deploy the Reference App!

Hands On in the Cloud

Next Up

- Workshop Session 6 will be scheduled for next week.
- Agenda:
  - Introduction to DevOps.
  - Next Steps / Wrap Up.
  - Troubleshooting.
  - Q/A.

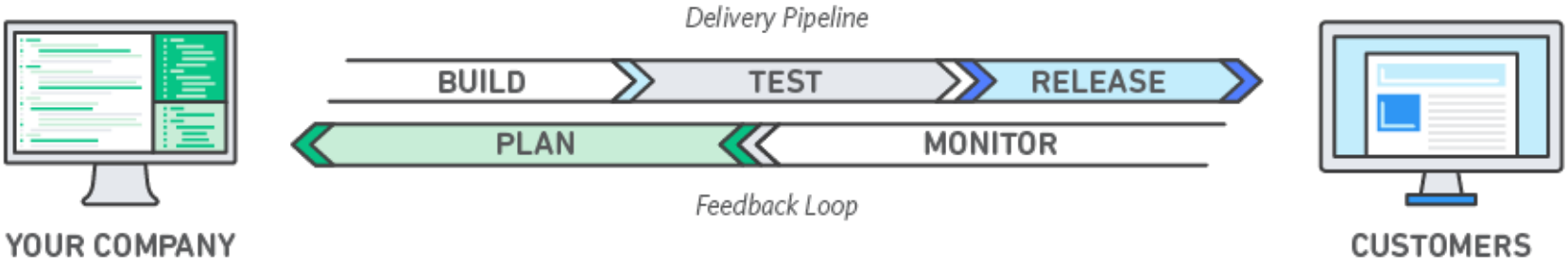
Workshop Session 6

Workshop Session 6

- Agenda:
  - Introduction to DevOps
  - Next Steps
  - Troubleshooting Issues
  - Q/A

What is DevOps?

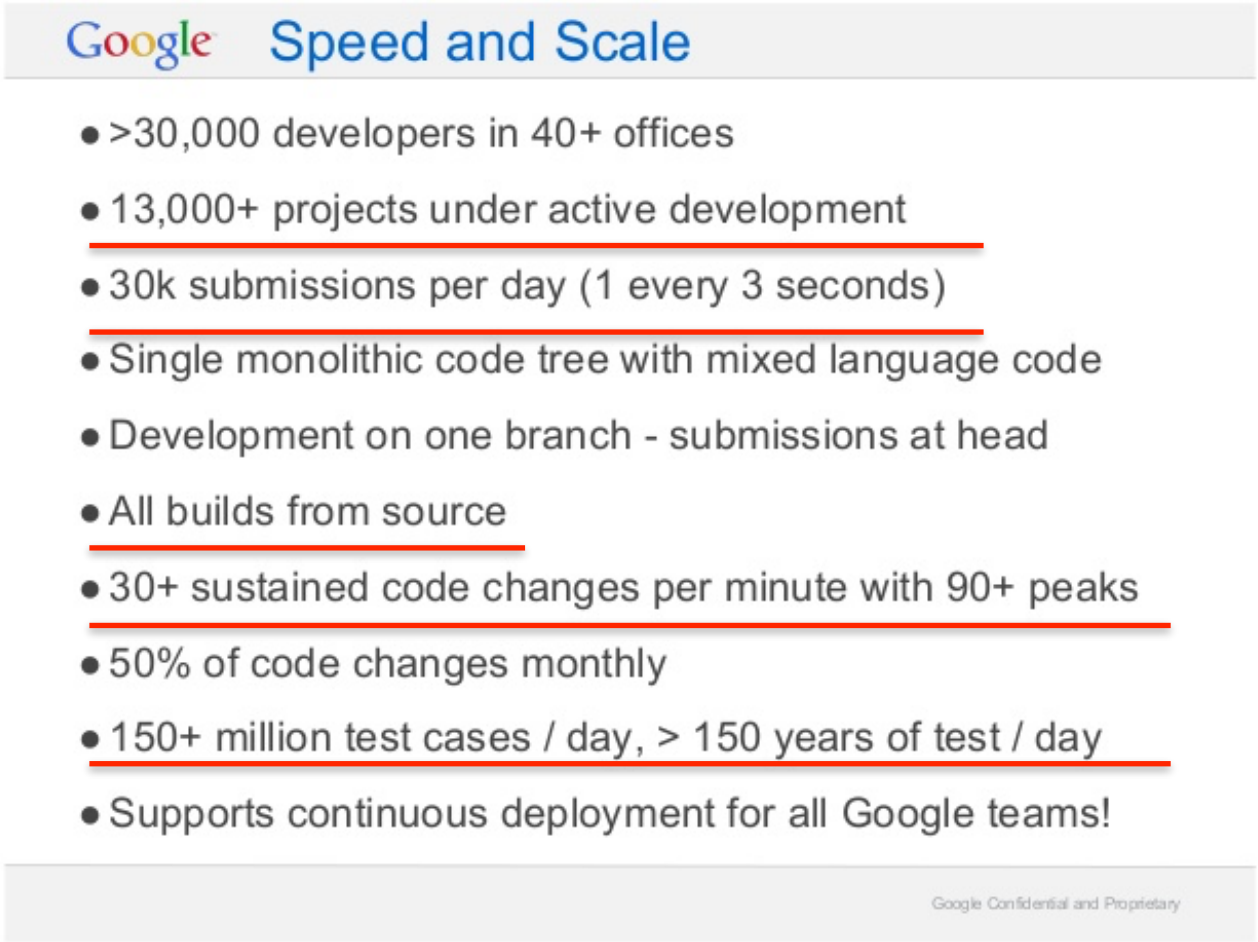
DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



Reference: https://aws.amazon.com/devops/what-is-devops/

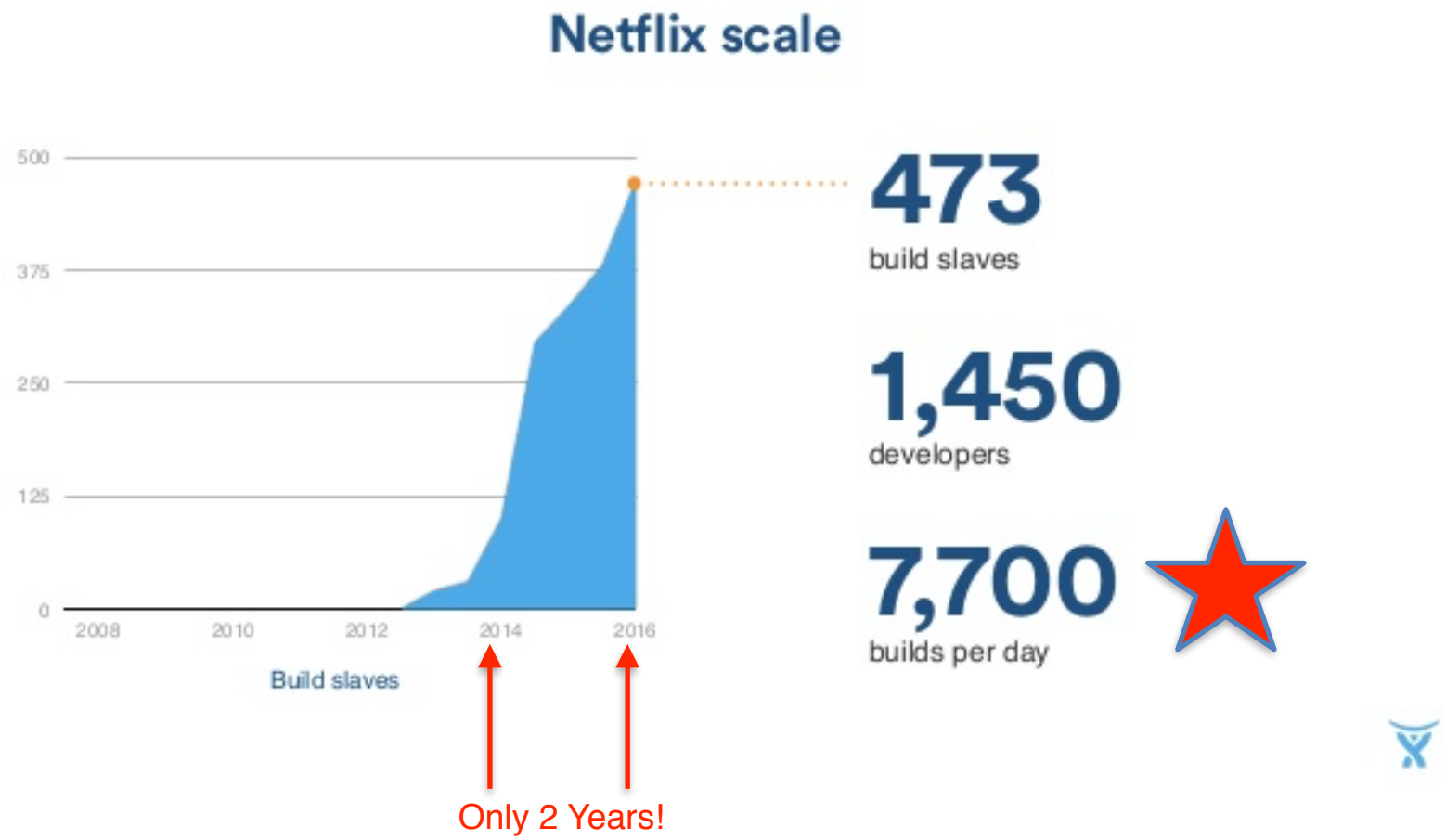
Think automation!
Netflix, Google, Amazon, and Facebook are leading the way.
There's more to software development than coding!

Can you imagine this?



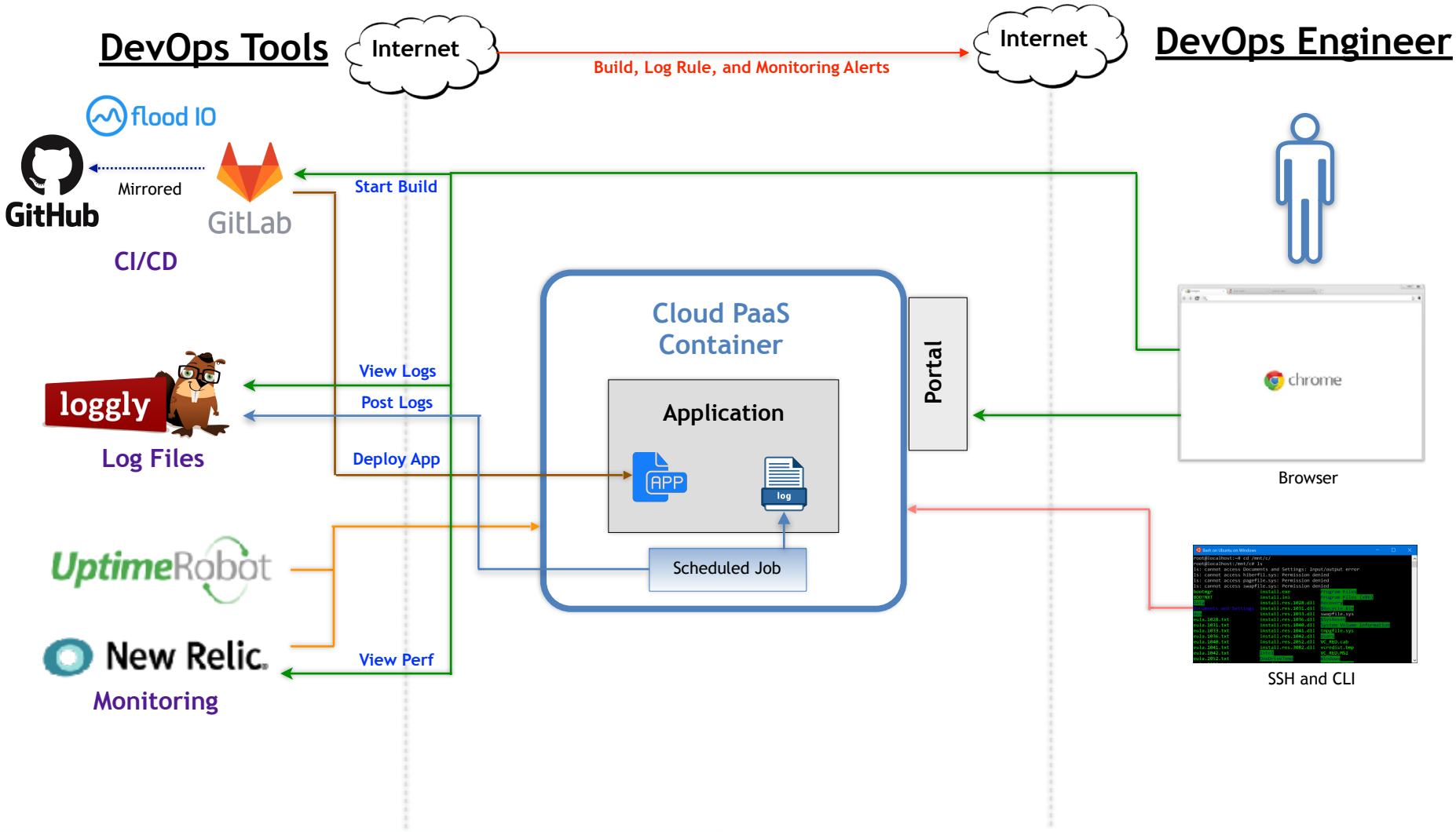
Learn these automated development and testing tools!

Or can you imagine this?



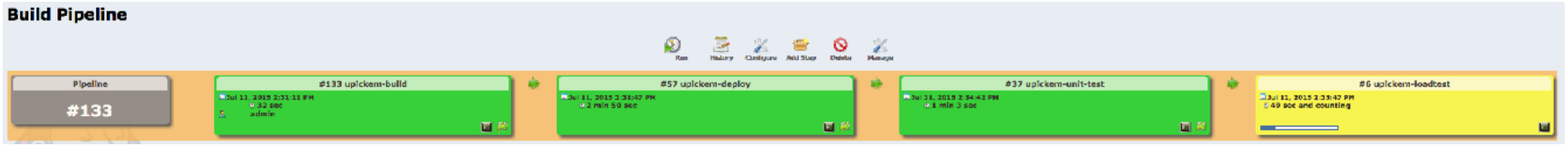
Automation is how these companies pull this off!

DevOps Reference Architecture



Cloud Powered CI/CD Pipeline

- Automated Jenkins or GitLab CI/CD Pipeline to build, deploy, execute unit tests, code coverage, and execute a load test using a dedicated OpenShift Cloud QA Test environment.
- Manual or automated deployment to OpenShift Cloud Production environment once all unit tests and functional tests pass in OpenShift Cloud QA Test environment.



Cloud Powered Code Coverage

Each build in the Automated Jenkins or GitLab CI/CD Pipeline includes running a Unit Test Suite along with validating that Code Coverage metrics are still being met.

Coverage Report - All Packages

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	36	61% 1375/2238	55% 343/618	2.552
com.onthreedgesc.nflpool.beans	20	77% 376/488	67% 19/28	1.079
com.onthreedgesc.nflpool.business	3	55% 337/612	45% 135/296	4.574
com.onthreedgesc.nflpool.data	3	49% 236/477	43% 28/64	4.07
com.onthreedgesc.nflpool.exceptions	4	25% 4/16	N/A N/A	1
com.onthreedgesc.nflpool.util	5	61% 153/250	53% 28/52	3.091
com.onthreedgesc.nflpool.web.service	1	68% 269/395	74% 133/178	9.538

Report generated by Cobertura 2.1.1 on 7/12/15 6:24 AM.

Cloud Powered Load Testing

Each build in the Automated Jenkins or GitLab CI/CD Pipeline includes running an N user JMeter Load Test using the Flood IO Cloud Load Test environment.

Cloud Powered Monitoring

- Get notified via email whenever any of your (Cloud) Servers goes down. One Service to achieve this is Uptime Robot.

- Measure and monitor critical business transactions and the performance of your JVM. One Tool to achieve this is New Relic.

Powerful Services and Tools are a necessity for DevOps!

What's Next?

If this workshop is well received we will be offering this workshop plus the following as an Explore More Session!

Domain	Details
Java	JUnit - Unit Testing Framework.
Java	Cobertura or Jacoco - Code Coverage Framework.
Java	PMD or Findbugs - Static Code Analysis Framework.
Java	JMeter - Performance Load Testing Framework.
DevOps	Jenkins or GitLab - Build Server.
DevOps	Jenkins or GitLab - CI/CD.
DevOps	Uptime Robot - Server Monitoring.
DevOps	New Relic - Application Monitoring.
DevOps	Flood.io - Cloud based Java Load Testing.

Questions?
What is not working?

Thank You

See you all in class or on campus!!