IBM Bluemix GitHub Repo Structure

8/31/15

v1.3

The following is a guide for structuring the GitHub repos for IBM Bluemix sample applications. The driving force behind this is an effort to standardize our open-source content to create a more cohesive brand experience across demos. Note that at the moment, these are evolving and fluid guidelines. There is continuous development being done on some of the artifacts mentioned in this guide and as they are updated, so too will their corresponding sections in this document.

Directory

Value Add

Any demos placed within the IBM Bluemix GitHub organization need to be reviewed to ensure they add value to the overall repository. This is needed to guarantee superior quality projects and to discourage the flooding of the repo with duplicate content. Optimally, this review will be conducted before development on the application has even begun.

Examples of value-add include (but are not limited to):

- "How-to" for using a community buildpack with no prior usage in the org
- "Hello World" demonstration of using a service with no prior sample
- Enterprise pattern reference architecture
- Promotional value for Bluemix partner (i.e. Vaadin, New Relic, etc.)

Sample Types

Through user research, we have concluded that our target personas are looking for distinctly different types of sample applications. It is important to recognize which category your app falls into when designing, developing, and promoting it. Here is how it breaks down in order of increasing complexity:

Hello World
Target Persona: Developers
These apps are simple, straightforward implementations of a Bluemix service or core feature. They are strictly meant as a reusable asset for developers. No "Deploy to Bluemix" necessary.

Sample App
Target Persona: Developers/Solution Architects
These are the apps that exhibit Bluemix use cases, with 1-3 services and a beautiful front end. Blog posts with tutorials normally accompany these and they are used to show off at conferences and large external events.

Demo
Target Persona: Tech Sellers/Developers/Solution Architects
Fully fleshed out samples that showcase vital enterprise functionality within a Bluemix app or multiple apps. They generally have accompanying scripts for pitching to enterprise clients with live-coding portions.

Reference Architecture Apps
Target Persona: Tech Sellers/Solution Architects
Built by the Guided Experience and Reference Architecture teams, these apps demonstrate enterprise pattern solutions using Bluemix. They generally involve multiple apps and services communicating with each other using various protocols in order to make up a larger system.

README.md Structure

In order to bring together the projects within our organization, we will need to standardize the primary repo documentation file: README.md. By making these files uniform across the org, we give off the impression that our sample apps are well thought out and collaborative as opposed to one-off efforts.

Keep in mind that a lot of the content in this README might be found elsewhere (i.e. documentation, tutorials, etc.). Avoid creating duplicate content where possible, by linking to other resources that would be defined as the source of truth.

Remember that this is only a guideline, not a rigid structure. The included sections should reflect what type of sample application is being delivered. If any of the following sections do not apply, feel free to omit them from your README. That being said, your README structure should include a subset of the following:

- **Overview** - This section includes the following:
    - Explanation of the application and its purpose. Answer the questions:
        - Why is this sample app relevant to the target persona?
        - What purpose does this sample app serve?
        - What does the sample app do?
    - Services that are used and why

    - Travis CI, Codeship, and other standard GitHub badge
- **Application Requirements *(if any)***: List any requirements for using the app, like browser and version. Include an architecture diagram if the app is anything more than a Demo.
- **Run the app on Bluemix**: If the sample app is anything more than a "Hello World" include instructions on how to deploy a new instance to Bluemix. This should be done by giving either:
    1. A Deploy to Bluemix button. The button itself is included by putting your Git repository URL in your `package.json` file and placing the below code into your README markdown. Currently, only CF apps are supported, with container and VM support coming in the future. If any additional steps are required to set up the app after using the D2B button, please note these as well.
        - `[![Deploy to Bluemix](https://bluemix.net/deploy/button.png)](https://bluemix.net/deploy)`
    2. Instructions on how to deploy the app to Bluemix after cloning the repo. Assume little to no prior Bluemix experience.
        - Needs to have sanity checks at about every 20 steps
        - Think accessibility first. If any tools are used (i.e. the CF CLI), briefly explain what the tools are for and how they are used.
        - Avoid directly referencing Bluemix UI components here; doing this helps avoid ACE changes invalidating your README
        - This section tends to be large and/or subject to changes. In order to limit the changes needed for your README, you can choose to:
            - Embed steps in another Instructions.md file and link there
            - Include a link to a tutorial hosted on a separate site (i.e. developerWorks or Bluemix Docs) instead
- **Run the app locally**: If applicable, include instructions on how to run the app on a user's local machine. Recommended for Demos and apps of similar to lower complexity.
- **Decomposition Instructions**: Instructions on how a developer/architect would take the sample application and extract the relevant code for reuse. For example, if the app is a "Hello World" to demonstrate integration of a specific service, explain what code is relevant to this integration and how one would reuse it.
- **API Documentation**: If the sample app exposes an API, either include basic documentation, link to a Swagger/Apigee API, or something similar. *Do **not** repeat API docs that are already published elsewhere*.
- **Contribution**: Note how you would like developers to contribute and/or log issues to your project. Also give steps for making pull requests. If your contribution directions become lengthy, break them out into a file called 'CONTRIBUTING.md' and link to this. Use the Puppet CONTRIBUTING.md file

as a reference. On the contrary, if your app is static and you do not want the community to contribute, note this and that all pull requests will be rejected.

- **Privacy Notice (CF-only)**: If using the deployment tracker, a privacy notice is needed to let the user know we will be collecting deployment data. The full privacy notice is in markdown below:
    - The YOUR_APP sample web application includes code to track deployments to Bluemix and other Cloud Foundry platforms. The following information is sent to a [Deployment Tracker] (https://github.com/cloudant-labs/deployment-tracker) service on each deployment:
        - Application Name (`application_name`)
        - Space ID (`space_id`)
        - Application Version (`application_version`)
        - Application URIs (`application_uris`)
    - This data is collected from the VCAP_APPLICATION environment variable in IBM Bluemix and other Cloud Foundry platforms. This data is used by IBM to track metrics around deployments of sample applications to IBM Bluemix. Only deployments of sample applications that include code to ping the Deployment Tracker service will be tracked.

    Lastly, include steps on how a user would remove the deployment tracker from your application.

Sample README: Feel free to use the Personality Box README as a guideline.

The GitHub community is very opinionated on README structure and content. The overall sentiment seems to be that the README should include everything a developer would want to know about what the project is and how they can use it. The co-founder of GitHub, Tom Preston-Werner, went so far as to suggest programmers should write their README first. This is a very similar notion to design-driven development.

Legal Compliance

License.txt – Each repo needs a license file that covers what license the accompanying source code is covered by. IBM approved licenses are the Apache 2.0 and MIT licenses.

Notice.txt – Each repo needs a file called 'Notice' with the following text:

```
This product includes software originally developed by IBM Corporation
Copyright 2015 IBM Corp.
```

Headers – Each source code file with more than 10 lines of code needs to include the following header at the top of the file:

```
/**
* Copyright 2015 IBM Corp. All Rights Reserved.
*
* Licensed under the Apache License, Version 2.0 (the “License”);
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      https://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an “AS IS” BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

Be sure to replace `Apache License, Version 2.0` with the license that you are using, if different.

Dependencies - Do not included dependencies (i.e. node.js packages) in your repository to avoid licensing issues. Always declare dependencies in such a way that the user is able to download them using dependency management. If you absolutely need to include dependency source code in your project, ensure they are using acceptable licenses.

Design

To aid the effort in building consistency across our apps, developers should follow design practices laid out in the Bluemix Pattern Library. The library includes design principles and UI components used across the Bluemix brand. Of course, there are exceptions to using elements in the pattern library; we want our apps to look unique in their own right. Just be aware that the more synchronous our sample apps are, the more we deliver a seamless experience across our sample app portfolio.

Note: As of this update, the pattern library is not yet a finished product. Expect ongoing development from the Bluemix FED team and use it accordingly.

Architecture Diagram

If the proposed sample app is meant to serve as a “Demo” or anything more complex, an accompanying architecture diagram should be included. This will make the application accessible to personas other then software developers. The diagram should also be included in the README.md file as a way of exposing it without requiring users to dig into the repo’s files.

In an effort to standardize Bluemix architecture diagrams, the Guided Experience team has put together all necessary iconography. Please use the icons and diagram elements located at this Reference Architecture Design Blueprints page.

For any questions on these architecture diagrams, contact Rick Osowski (osowski@us.ibm.com) or Angela Runge (arunge@us.ibm.com) for more information.

Marketing Copy (large apps only)

These sample apps will be extremely useful to developers and architects alike, but it is important to first get them engaged and create awareness with short-form marketing copy. This can come in the form of a short YouTube video or a succinct blog post on DeveloperWorks. The choice is yours. Just remember, the reach of your sample app often depends on the quality and appeal of your promotional content.

The Bluemix developer and community building lead, Dan Kehn (kehn@us.ibm.com), can help advise on what content may be suitable for your particular application.

Deployment Tracker

In order to facilitate the measurement of success of our sample apps, the greater developer advocate team has built a tool to track the deployment of our Cloud Foundry sample apps. See the following GitHub repositories for instructions on how to implement the deployment tracker into your:

- Node.js App
- Java App

It is also required that you include the privacy notice in your README when you implement the deployment tracker.

This is currently only available for Node.js and Java CF applications. Support for other languages and compute (Containers/VMs) will be coming in the near future.

Deploy-to-Bluemix (D2B) Button

Applicable projects should be instrumented to allow a user to deploy their application to Bluemix in as simple a manner as possible. To enable this, we are supporting the Deploy-to-Bluemix button built by the DevOps team. This helps us get more people onto Bluemix as well as track how many times each app is being deployed.

The D2B button should be included in the README file, as well as any blog posts/tutorials written on the sample app.

Discovery JSON Document

We would like to expose content to users within the product at relevant times. More importantly, we would like any application utilizing the GitHub API to be able to identify relevant Bluemix sample apps. To facilitate this discovery, we plan to include an additional JSON document in the root of the project folder that describes the relevance of the project. The proposed structure of this `discovery.json` is as follows:

- **discover *(boolean)***: denotes whether the project should be considered by the content microservice

- **type *(string)***: degree of complexity [hello_world, sample_app, demo, reference_architecture]
- **compute *(array of strings)***: types of compute used in the demo [cloud_foundry, docker, openstack]
- **runtime *(array of strings)***: runtimes that the app uses [java, node, python, ruby, php, go, aspnet]
- **services *(array of strings)***: list of services used by the app [See below for service names]

```
{
  "discover" : true,
  "type" : "sample_app",
  "compute" : [
    "cloud_foundry"
  ],
  "runtimes" : [
    "node"
  ],
  "services" : [
    "WatsonPersonalityInsights", "Cloudant", "Box"
  ]
}
```

CoreMetrics Links

In an effort to identify and measure all traffic that we drive to Bluemix, we will make sure that all links use CoreMetrics tags. All links need to be tagged using the cm_mmc URL query parameter, which is a 4-part string, delimited by "-_-". The following is the structure we will use for our links:

https://console.ng.bluemix.net/?cm_mmc=Display-<Channel>-_-BluemixSampleApp-<AppName>-_-<Runtime>-<PrimaryService>-_-BM-<Org>-<Country>-<City>

Note: Country and City tags are optional

This format lets us sort by channel, effectiveness of plugs across an entire sample app, determine popular runtimes and services, and track engagement by city. The actual tags are located below.

How to View Raw Data

1. Obtain 'Basic Access' for Datameer
    a. View the Datameer Info Release Blueprints page for instructions on how to request access: https://releaseblueprints.ibm.com/display/CLOUDOE/Datameer+Information

    b. Justification: View metrics for Bluemix CoreMetrics data for my sample applications.
2. Open up Datameer at https://9.39.221.55:8443/login
3. Navigate to: Users -> jbsoloyer@us.ibm.com
4. Open up the 'SampleAppCoremetrics_Public

This table is currently a compilation of all CoreMetrics data with the 'BluemicSampleApp' tag. As we progress and learn more about this data, we will create visualizations around the important data points.

Channels

- GitHubReadMe
- DeveloperWorksTutorial
- SampleAppLink

Runtime

- Java
- Node
- Python
- Ruby
- PHP
- Go
- ASPNET

MainServices

- Watson
  - AlchemyAPI
  - WatsonConceptExpansion
  - WatsonConceptInsights
  - WatsonLanguageIdentification
  - WatsonMachineTranslation
  - WatsonPersonalityInsights
  - WatsonQuestionAndAnswer
  - WatsonRelationshipExtraction
  - WatsonSpeechToText
  - WatsonTextToSpeech
  - WatsonTradeoffAnalytics
  - WatsonVisualRecognition
  - CognitiveCommerce
  - CognitiveGraph
  - CognitiveInsights
- Mobile
  - AdvancedMobileAccess
  - MobileCloudant
  - MobileAppSecurity
  - MobileData

    - MobileQA
    - MobileQualityExtensions
    - PresenceInsights
    - MobilePush
    - MobilePushiOS8
    - Twilio
- DevOps
    - AutoScaling
    - DeliveryPipeline
    - MonitoringAndAnalytics
    - TrackAndPlan
    - BlazeMeter
    - LoadImpact
    - NewRelic
    - IBMGlobalization
    - IntegrationTesting
- Web and App
    - BusinessRules
    - DataCache
    - MQLight
    - SessionCache
    - Workflow
    - WorkloadScheduler
    - Box
    - CloudAMQP
    - DreamFace
    - Geocoding
    - MemcachedCloud
    - NamaraIo
    - Reappt
    - ReverseGeocoding
    - SendGrid
    - Simplicite
    - Statica
    - TravelBoundary
    - Ustream
    - ValidateAddress
    - DocumentGeneration
    - Gamification
    - RabbitMQ
    - Redis
- Integration
    - APIManagement
    - CloudIntegration
    - SecureGateway
- Data Management

    - Cloudant
    - DataWorks
    - ObjectStorage
    - ObjectStorageV2
    - SQLDB
    - ClearDB
    - ElephantSQL
    - MongoLab
    - RedisCloud
    - MongoDB
    - MySQL
    - PostgreSQL
    - ProbabilisticMatch
- Big Data
    - BigInsightsHadoop
    - DashDB
    - GeospatialAnalytics
    - IBMAnalyticsHadoop
    - TwitterInsights
    - TimeSeriesDB
- Security
    - AppSecurityManager
    - AppScanDynamicAnalyzer
    - AppScanMobileAnalyzer
    - iOSMobileAnalyzer
    - SSO
    - StaticAnalyzer
    - aPersonaASM
- Business Analytics
    - EmbeddableReporting
    - PredictiveModeling
    - CupenyaInsights
- IOT
    - IoTFoundation
    - FlowthingsIO
    - IoTWorkbench