DevTreks –social budgeting that improves lives and livelihoods

Cross Platform Containers

Last Updated: June 10, 2016 First Released: March 03, 2016

Author: Kevin Boyle, President, DevTreks

Version: 2.0.0.beta1

A. Introduction (1*)

Containers offer a way to run software as microservices or web api data services. The following definition applies to these types of data services (Fussell, 2016):

Microservice applications are composed of small, independently versioned, and scalable customer-focused services that communicate with each other over standard protocols with well-defined interfaces.

These types of services typically run in a software containers which do not interact with host computers in a way that could disrupt the configuration of that host. The container interacts with the host's core operating system, but not in a way that disrupts anything on the host (other than memory, cpu …). They also spin up fast –meaning that servers holding the containers can be quickly added to server farms to handle spikes in traffic.

The following sections discuss additional details about Containers. Section G discusses design goals for Version 2.0.0 that address shortfalls mentioned in these sections.

B. Licenses

Nothing is free in this world (especially to economists), including containers. Most of the software deployed in containers is subject to the same software license requirements as deployments on regular computers. That includes operating systems and databases –the container is considered a separate, stand-alone computer. Because of this restriction, the examples in this reference focus on enterprise-level containers –containers that are deployed by organizations who own enterprise-wide licenses that allow them to deploy software on new computers, including containerized computers.

Because not everyone can afford multiple software licenses, especially potential customers in developing counties (or for that matter, the author), we may continue releasing localhost packages.

C. Host Computers

The computers that host containers must be configured for that purpose. For example, the first example demonstrated below is configured to host containers on Microsoft Azure cloud computing data center servers. Once configured, the host computer can then deploy the required container. Version 2.0.0 is currently being refactored with the goal of supporting Windows, Linux, and Mac, containers.

A lot of containers are available. The author recently viewed an interview with an employee of Docker (one of the first companies to offer container software) who stated that 80,000 containers are available for download from their "hub" site. As this technology matures, many more containers can be anticipated to be available. The burden of deploying the first container can be lessened by carefully planning which other containers may be needed in the future.

D. Source Code

The container technology that DevTreks will use is still in "release candidate" status. An official DevTreks container release will not take place until after the official release of the Net Core 1 technologies. Source code users should be actively experimenting with the same technologies. The source code started being added to github with Version 2.0.0.beta1. A lot is happening in this field and this is just one of several new or upgraded technologies that will be included in the next several source code releases (i.e. Sql Server 2016, Entity Framework 7 (EF7), Net Core 1.0, AspNet Core 1.0, Azure 2.9, Azure App Service web app, R and Python upgrades).

Further details about the source code can be found the Source Code reference in the Deployment tutorial.

E. Subscriptions and Public Goods Software

Containers may make it much easier to deploy subscription-based software. That's very important to software companies, because technology is advancing at a rapid pace. They must hire talented software developers (i.e. the kind interested in working on climate change or malnutrition) to stay apace with the technology.

Although the DevTreks focus on public goods software may sound contradictory with fee-based subscriptions, the reality remains that many of the conventional institutions responsible for public goods delivery, don't have a good track record in using technology to deliver services more efficiently. And non-conventional companies who do understand how to deliver public goods services more efficiently, dislike convention. Especially the convention of rent-seeking. Even software companies that focus on public goods need reliable revenue streams that don't rely on conventional-institution funding. Subscription-based software may be their best option. Recent voting, and political campaign contribution, patterns in the U.S. suggest that younger generations, in particular, may be logical supporters of companies that specialize in this business ethos (including those who can only afford "$28" or less).

F. Version 2.0.0 Release (2*)

Version 2.0.0 is currently being refactored with the goal of supporting Windows, Linux, and Mac, platforms. The real goal is ensure a sound future for the source code. By complying with the objectives of the development tools, Microsoft's Net Core 1 open source direction, all of the improvements in those tools become available for immediate and future use, including containers delivering microservices.

As of the release date, enough tests have passed to justify deploying the product on Azure and to add the source code to github. Plenty of work can still be done, but the software passes an appropriate bar for a small ngo –it works. Appendix A goes into greater detail about what's been done and what remains to do.

The refactor has introduced new bugs, so testing will have to take place over an extended period. The tradeoff between new bugs in the short term and better software in the long term is worth the added work. But hiccups can be anticipated on this design path.

If these goals are achieved, subscription-based software becomes a much more probable business model. Especially to organizations who want to support the goals of small, non-conventional, nonprofits that specialize in public goods software (whether this shop or your shop).

G. Examples

The following examples demonstrate common scenarios for deploying containers containing DevTreks.

Example 1. DevTreks Container deployed in one public Azure cloud computing data center

-under planning for Version 2.0.2 or Version 2.0.4

Example 2. DevTreks Container deployed in one private network (i.e. Nano Server or Windows Pro 10 network group)

-under planning for Version 2.0.2 or Version 2.0.4

Example 3. DevTreks and DevTreks Container deployed on Linux and/or Mac platforms

-under planning

Example 4. DevTreks Container *Subscription* deployed in one public Azure cloud computing data center

-under planning (read Section F)

Summary

Containers delivering API-based data services may take Conservation Technology Assessments (CTAs) a step closer to being a best practice technology for tackling climate change, malnutrition, agricultural development, public infrastructure investment, food system health, health care efficiency, and other societal issues where money is scarce and needs to be spent “economically”. They may also may make it easier for people to make decisions that improve their lives and livelihoods.

References

Fussel, Mark . Why a microservices approach to building applications? Microsoft, 2016. Last accessed April 29, 2016 at:

https://azure.microsoft.com/en-us/documentation/articles/service-fabric-overview-microservices

Microsoft. ASP.NET 5 Documentation. Release. Version dated April 27, 2016. (To be renamed ASP.NET Core 1)

Microsoft. Entity Framework Documentation. Release 7.0.0. March 11, 2016

Footnotes

1. This explanation of containers reflects the author's evolving understanding of containers (which will continue to evolve). Although a refactored ASP.NET Core 1.0 application might just be dropped into a container, and voila, version 2.0.0, serious CTAs aren't produced using "voilas". So, time to read Footnote 2 again and "care, think, and act" appropriately. The full challenge of refactoring a multitier database application to containers delivering microservices and/or web api-based services isn't known yet. Logic says to try. In the meantime, reality says to release Version 2.0.0 with a fully working ASP.NET Core 1.0 multitier database application.
2. One single software developer should not be expected to produce a completely stellar product. The Calculators and Analyzers reference discusses serious limitations with the current business model employed by DevTreks and other public goods-oriented software development ngos. An important lesson from this project may be to teach people how to care, think, and act, like a technologist. Tough societal problems can be, and should be, tackled with concrete public goods software. The time for talk and papers is over –it's time for action and source code.
3. Json versus xml is not considered an issue because clients only receive stateful html. The servers process the xml, or json, into html. How that html is generated is not considered critical, provided that the html conversion performs well and that the all of the data

follows w3.org protocols (i.e. for html, xml, or json). Html conversion and management will always be an area that can be improved in web applications.

Improvements, Errors, and New Features

Please notify DevTreks (devtrekkers@gmail.com) if you find errors in these references. Also please let us know about suggested improvements or recommended new features.

A video tutorial explaining this reference can be found at:

Won’t be available until after testing on Azure takes place.

Appendix A. Refactor Status

The following lists the current status of the refactor.

Completed as of the release date:

1. The Services layer has been moved entirely into the UI layer, enabling them to be “provided”, or least used, as standard UI services. All html manipulation has been removed from the data layer and consolidated in the UI layer.
2. MVC 5 html manipulation has been upgraded to MVC 6 techniques. Some of the MVC 6 techniques, such as greater use of compiled classes, are notably better than some of the older techniques. A lot of html manipulation code has been eliminated or consolidated into 2 classes that implement 1 uniform html extension pattern.
3. Club management has been simplified and strengthened. Most edits now take place using standard MVC and EF7 edit patterns, rather than some of the previous custom patterns. Localizations no longer rely on linked calculators -localization properties can be edited directly on a Local html form. The new localization pattern deprecated using drop down lists to select units of measurement in base element html forms –the author experienced just too many missing units over the years. The Social Budgeting tutorial has been upgraded with the refactored club management (i.e. everything on the club panel has been refactored).
4. Security has been upgraded to the default AspNet Core 1 security. Login management is noticeably improved.
5. The database has been upgraded to Sql Server 2016 RC1.
6. The data layer models have been “scaffolded” to Entity Framework 7 RC1 models. The goal is full support for future data model changes via migrations. The EF7 upgrade required substantive changes to the EF models and their insertion, update, delete, and find, patterns.
7. The data layer, in particular, implements much needed Core 1 approaches for getting and setting all configuration, route, and httpcontext, settings. For example, db connection strings are all uniformly retrieved using a single line of code in a single function.

8. Performance, such as the loading of several common html views, has been increased, sometimes by an order of magnitude. Some of these improvements came from finally taking the time to fix known, but low priority, UI bugs (i.e. double controller hits, videos that load unnecessarily in some browsers) and speeding up html loading.
9. The MathNet library has become the official library used by algorithm 1 because all the associated subalgorithms, in fact, rely on it. All of those subalgos can be improved, but good cause may exist to put priority on adding Intel Math Kernel and NumPy as "algorithm libraries". The CTAP reference documents that this refactor will not happen until Version 2.0.2 or 2.0.4.
10. The only table that had to be refactored so far, outside of the new AspNet Core 1.0 "login" tables, is the Local table. The table, along with the corresponding AccountToLocal EF model, had to be refactored to support a simpler localization pattern for club management. The same table, models, and dbcontext, must be refactored in existing databases.
11. The current refactor has been successfully deployed to localhost and successfully debugged using storage for both localhost's file system files and Azure's blobs.
12. The 3 features that strayed most from standard content management practices, Story-telling, DevPacks, and Package/Zip, have been refactored. Their refactor is documented in the Story Telling, DevPacks, and Calculators/Analyzers tutorials. These features are typically debugged last.
13. The code has been refactored using the Release Candidate 2 Net Core libraries. This refactor only uses the Net Framework 4.5.1 libraries referenced as "net451". The code has not been refactored to be 100% compatible with the Net Core 1 libraries referenced as "imports dnxcore50". The primary difficulty is a System.Xml.Xsl library "missing" from the core 1 libraries. Once refactored, including the missing library, DevTreks should be compatible with Linux and Mac servers, but tests will be needed for confirmation.
14. Extension scripts for copying dlls stopped working in beta2 and were replaced with ContentViewModel code. RC2 conventions such as buildOptions either didn't work or were not fully undersood. The new code requires using the release config in Visual Studio to log in and then previewing data. The site can then be published and deployed.

To do for Version 2.0.0:

1. The Azure refactor requires setting up Azure to support the new code base and updating the Sql Azure database. A new video tutorial will be added to the Deployment tutorial after the Azure site is debugged.

To do for Version 2.0.2 and Version 2.0.4:

2. It's likely that the full refactor of the Azure site will involve microservice containers and/or web api-based containers. These technologies appear to be the logical future for cloud and mobile software development. Although they have a learning curve that increases the riskiness of producing a flawed product, some cause for optimism can be found in DevTreks design:
    a. DevTreks main content is delivered using concrete Services -Inputs, Outputs, Operations, Components, Outcomes, Operating Budgets, Capital Budgets, DevPacks, Linked Views, and Resources. Delivering those "microservices" through public, programmable, APIs is considered good design. In fact, they are already delivered through a data repository API.
    b. Extensions, which are calculator and analyzer dlls, use Managed Extensibility Framework (MEF). MEF, microservices, and web apis, all employ loosely coupled services that interact with other services through well-defined "interfaces and protocols", such as URLs and APIs.
3. Html upgrades are ongoing work. Some of the UI interfaces can be improved.
4. Each additional feature in the new security and login pattern, such as 2 factor authorization, has to be reviewed and tested. This will be put off until 2.0.2 or 2.0.4.
5. The EF7 refactor appears stable but needs more testing.
6. Code improvements are ongoing work.

7. Current R and Python statistical libraries will be upgraded, possibly to the new server versions of these products (i.e. R Server). The CTAP reference documents that this refactor will not happen until Version 2.0.2 or 2.0.4.
8. The database has been refactored to support the use of database scripts, or EF7 migrations, for all db changes. It has not been refactored to support the goal of annual changes via scripts, or EF7 migrations, for structural changes only –not for content. The biggest recurrent content changes result from stylesheets being stored in the db and in file/blob storage. When a stylesheet is updated, existing databases have to updated with the new stylesheet. The lead developer actually thinks that's sound storage –so alternatives are still being considered.