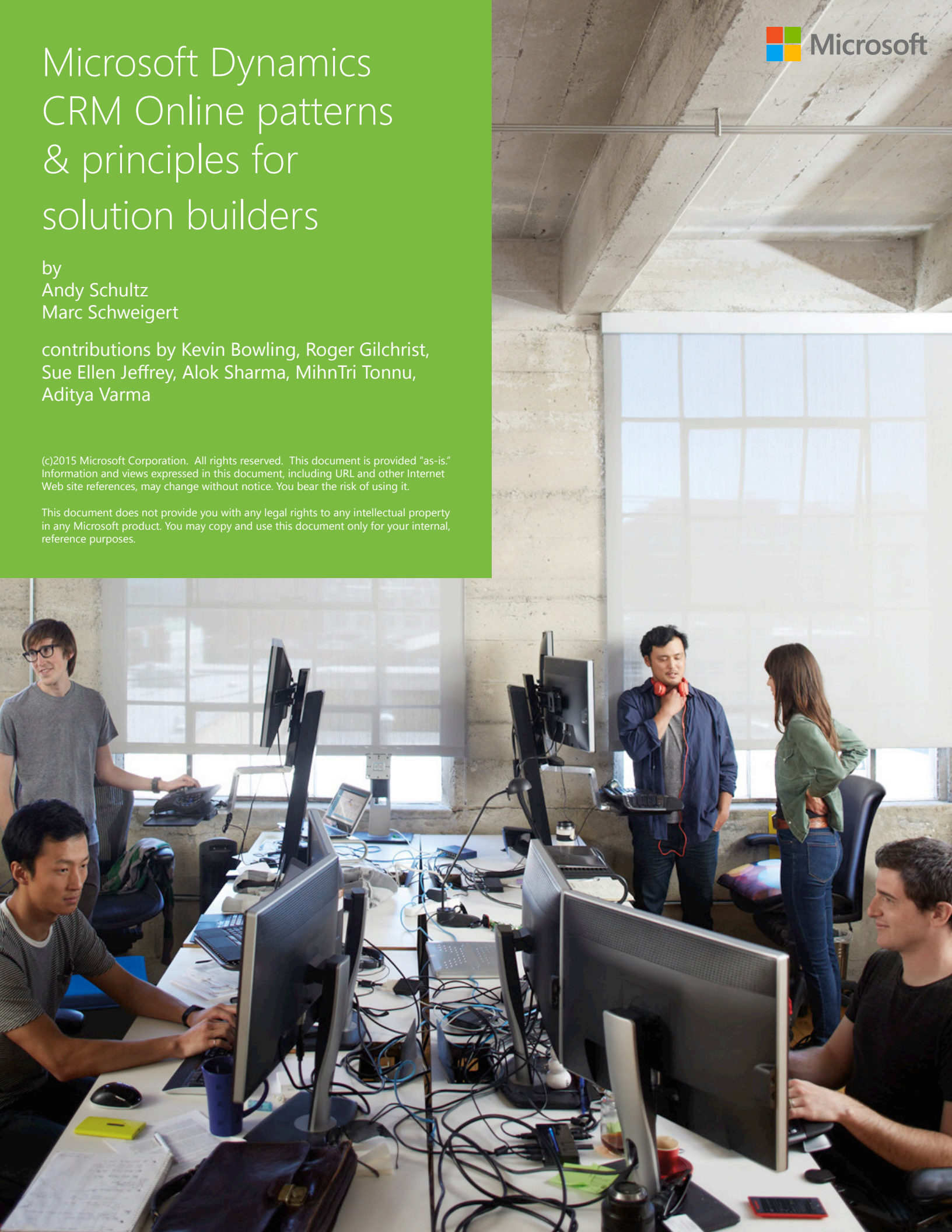
Microsoft Dynamics CRM Online patterns & principles for solution builders

by
Andy Schultz
Marc Schweigert

contributions by Kevin Bowling, Roger Gilchrist, Sue Ellen Jeffrey, Alok Sharma, MihnTri Tonnu, Aditya Varma



Introduction

At Microsoft, we hear customers in many industry verticals asking for cloud solutions. We're prioritizing these solutions, and communicating to customers the value of leveraging the Microsoft cloud. We have a highly available, world class data center infrastructure powering our cloud properties. This infrastructure delivers value that customers would have to pay orders of magnitude more money to create on their own. Customers can leverage the investment made by Microsoft to help them meet their demands and achieve the scale necessary for their businesses. Our high-value offerings like Microsoft Azure, Microsoft Dynamics CRM Online, and Microsoft Office 365 come with out-of-the-box interoperability, and enable deeper custom interoperability, providing capabilities much more quickly than an on-premises solution could achieve. And these cloud properties are enterprise-ready, operating at massive scale and with availability across much of the globe.

Microsoft Dynamics CRM Online is a crucial part of Microsoft's cloud strategy. While Azure provides services that allow infrastructure to be moved to the cloud, and Office 365 provides tools for everything from communication to document collaboration, Microsoft Dynamics CRM Online provides users the power to manage business data and processes related to people, places, and things. Hand in hand with Azure and Office 365, Microsoft Dynamics CRM Online completes the Microsoft cloud offering for businesses and governments who want to move their business computing to the cloud.

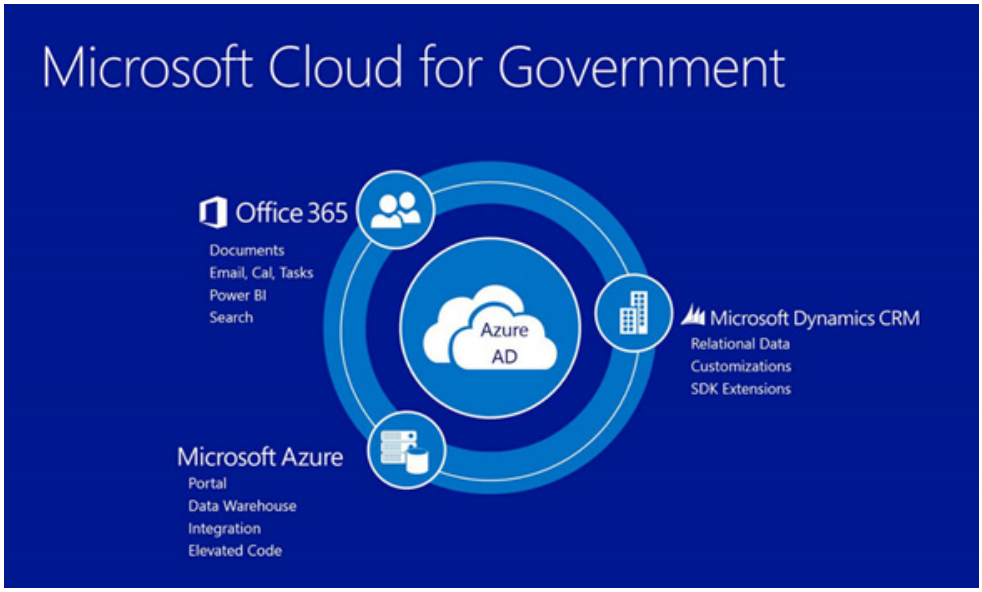
Customers are asking for Microsoft Dynamics CRM Online. If the solution you build doesn't run in Microsoft Dynamics CRM Online, you might find yourself on the outside of important opportunities in the near future as you're unable to meet customer requirements. Beyond just making certain "tweaks" to an on-premises solution or to your solution-building approach, designing a solution for the cloud requires you to change your philosophy. This guide is designed to help you understand this fundamental shift in mindset by outlining the key areas where cloud solutions require different thinking. Whether you need to change an existing on-premises solution to make it deployable in Microsoft Dynamics CRM Online, or you simply need to know how to design your next solution for the cloud instead of on-premises deployment, we encourage you to work through the following guidelines now, so you're ready when the time for action arrives.

The Value of Complementarity

Interoperability between cloud services is one of the strong added values such services provide. Microsoft cloud offerings are no different. Microsoft Dynamics CRM solution builders have traditionally designed solutions focused on customer relationship management (CRM), sometimes with another product added here or there to complement CRM capabilities and add a few more. To realize the value of the cloud for solution building, it's necessary to leverage the complementary services to which you have easy access. You should also understand that some of these should be seen as the cloud equivalent of or replacement for the on-premises resources you are accustomed to using, and others represent new ways of thinking and building solutions that have no on-premises equivalent.

In many instances, enabling online deployment of solutions that you've previously built for on-premises deployment will require more than just Microsoft Dynamics CRM Online, because you may have used on-premises resources that don't exist in the cloud – a web application running on IIS, integrated into the CRM UI, for example. To bring this full solution to the cloud, you'll need to use Microsoft Dynamics CRM Online and Microsoft Azure, with the latter hosting your web application.

But there's much more value in the complementarity between cloud applications than in just replicating what you've done on-premises. An expanding set of Microsoft Azure services, Office 365 services (like Delve, Sway, and Video), and more and better Office APIs create possibilities that didn't exist with applications installed locally or in a private data center. For instance, Office 365 includes SharePoint Online, which has out-of-the-box functionality for document storage and creation. This document interoperability leverages Office 365, which allows you to read or edit Office documents without ever leaving the browser.



Architecturally, Microsoft Azure Active Directory is a foundational component of a solution built on the Microsoft cloud. It provides a single identity for a user that works across Microsoft Dynamics CRM Online, Office 365, and custom applications you've built in Azure. It also provides the capability for a hybrid cloud solution and for single sign-on with an on-premises Active Directory deployment by offering the option to federate with Active Directory on-premises.

Many robust business systems require things like data warehousing, interoperability with external systems, and certain advanced coding scenarios. All of these things are available to users through Azure.

It's been wisely stated that a mobile application that simply takes something you used to do at your desk, and allows you to do that same thing walking around, is of little value. The true value in mobility is in letting people do things that weren't possible at all without mobile computing.[1] A similar principle is true with the cloud. Simply taking something that used to be on-premises and making it run in the cloud is of limited value. Using the complementary cloud services Microsoft puts at your fingertips makes it possible for users to do something that wasn't possible without the cloud. If you're mindful of the possibilities, you can build much more value into your cloud solutions. Keep the value of complementarity in mind as you read on.

The following sections discuss the areas you'll need to evaluate when building a solution for Microsoft Dynamics CRM Online (or converting a solution that was originally built for Microsoft Dynamics CRM on-premises). While it is likely that not every category will be relevant for every solution, this guidance is designed to help you begin thinking about how to design your solution for Microsoft Dynamics CRM Online – whether you have an existing on-premises solution or are starting from scratch.

I also recommend taking a look at the presentation from Microsoft Convergence 2014 Europe titled "Design for Cloud with Microsoft Dynamics CRM".[2] The presenters, Amir Jafri and Ian Smith, walk through many of the same principles covered in this paper, and you may glean additional insights by hearing them discuss these topics from a different perspective.

Cloud Principles

There are some principles that, while they don't translate directly into detailed rules for what you should and shouldn't do in Microsoft Dynamics CRM Online, still deserve serious consideration when designing this type of cloud solution. We'll start with these principles before we get down to the details.

[1] See Cyriac Roeding, "You Have to Break the Egg." Stanford University E-Corner. http://ecorner.stanford.edu/authorMaterialInfo.html?mid=3174

[2] http://convergence.evo-td.com/library/BKCRM406

Scale and Shared Resources

Microsoft Dynamics CRM Online instances use a pool of shared resources, such as web server, async service, report server, disk I/O, and compute power.[3] This is important to understand, because it has both an upside and a downside. We'll discuss the details of the downside first: no portion of these shared resources are dedicated strictly to the instance running your solution – they're shared. That means that you must design your solution to accommodate potential scenarios where these resources don't perform your requests immediately. For example:

- An asynchronous workflow doesn't run immediately. Asynchronous workflows are sent to the async queue on the async server, a shared resource. Your workflow job may have to wait its turn in the queue before it executes; although the execution time is often very quick, there's no guarantee that it will be.
    - If the impact of a workflow will be small enough, use a synchronous workflow instead of asynchronous to avoid the possibility of having to wait for the workflow job to complete. Synchronous workflows don't get sent out to the async server – they're processed immediately by the web server resources utilized by the user's session. When creating a workflow, therefore, you have to weigh the potential delay from the async service from an asynchronous workflow with the possible impact on performance from a synchronous workflow.
    - Also, be aware that while there are no strict limits on the amount of workflow jobs you can send to the queue, if you or any of your neighbors sharing the resource are using an inordinate amount of resources, you may have a governor placed on your usage.
- You need to be aware of database transaction volume. The database is the biggest bottleneck in both an on-premises deployment and Microsoft Dynamics CRM Online. Being aware of database transaction volume is important not only when you're importing data, but in typical CRM usage scenarios as well. Understand the way transactions lock the database and optimize your code accordingly. In Microsoft Dynamics CRM Online, the strict timeout limits mean you can't afford to have a transaction which waits for minutes while the database is locked for other transactions ahead in the queue.
- Regarding database performance, you should also understand that while you don't have access to the database to create your own indexes, you can submit a request to Microsoft Dynamics CRM Online to have (an) index(es) created for you.

Now, let's discuss the upside: In an on-premises deployment, your solution has access to exactly the amount of resources that your customer has allocated to it. In Microsoft Dynamics CRM Online, you don't have to worry about these shared resources - Microsoft invests in hardware to provide the necessary capacity for your workload. As the needs for your deployed solution grow, the Microsoft Dynamics CRM Online service can provide the resources to help meet those needs.

Also, it's important to be aware of the published limits that do exist in Microsoft Dynamics CRM Online, as well as the fact that you can get around these limits, if your customer requires it, by working with your Microsoft contacts and CRM Service Engineering. These limits include obvious things like storage space, entity count, and workflow count, as well as more obscure measurements such as the ExecuteMultiple resource governor that may slow down data migrations. If these published limits pose a challenge for a customer, you can work with Microsoft to meet your customers' needs.

Enforced Best Practices

Everybody has felt a little guilty when they wrote some bad code in the past. Well, in Microsoft Dynamics CRM Online, the temptation to write icky code has been removed in a lot of places, especially where it comes to resource utilization. The reason for enforcing certain good practices in Microsoft Dynamics CRM Online solutions is simple and, we hope you'll agree, legitimate. Certain types of customizations, written the wrong way, could adversely impact the performance and stability of

[3] If you want to understand the online service architecture in more detail see: http://convergence.evo-td.com/library/DDCRM407

the Microsoft Dynamics CRM Online service for other customers. In an on-premises environment, the customizations you write execute on resources you control. But because resources are shared in Microsoft Dynamics CRM Online, if your async plugin were able to take 15 minutes to execute, there would be less of the async service resource during that time for other customers' workflows and async plug-ins – which means your icky code is degrading the performance of other customers.

This is closely related to the last point about shared resources – but in this case, we're listing the things you're flat-out not allowed to do.

Plug-ins	2 minute timeout
SQL	30 second timeout for database transactions
Running workflow jobs	Fair use – no specific hard limits, but the resource is balanced across organizations
Direct database access	Not allowed

So what do you do if you need to do perform some logic that takes longer to execute? Clear your mind of that bad thought you were thinking – the answer is NOT to abandon plans to build for Microsoft Dynamics CRM Online and stick with on-premises forever. The answer is to put that code in the proper place in the cloud. Read on.

Use PaaS or IaaS in Azure when Necessary

Of course there are times when your data processing requirements will require more time and resources than are typically available in a SaaS application like Microsoft Dynamics CRM Online. No worries – you can run this code in Azure. By passing the code off to Azure, you remove the resource limits and gain the ability to configure the size and power of the resources that will run your code. When you do this, make sure that your Azure service runs in the same region as your Microsoft Dynamics CRM Online org to ensure minimal latency. Microsoft Dynamics CRM Online and Azure share data centers – which means the communication between your app and Microsoft Dynamics CRM Online can be super-fast if you put it in the right place.

Also, be aware of the fact that you can write code in Azure that performs single-sign-on with Microsoft Dynamics CRM Online. You can also give your code access to the Microsoft Dynamics CRM Online web service via the signed-in user (in which case, your code will run under the signed-in user's account).[4]

Configure Where Possible

Since we've been talking code, let's now talk about proper restraint. Don't write code just because you know how. Consider the long-term maintenance requirements, and use the declarative tools in CRM for building business logic when possible. Configurations are easier for others to analyze and understand without having to open up Microsoft Visual Studio. They are also a more integrated part of the CRM solution packaging model, which means that they'll be checked for compatibility during solution import – which is very helpful given the frequency with which Microsoft Dynamics CRM Online is released (approximately every 6 months). And while it's easy to write code that doesn't follow Microsoft's guidelines for staying supported, it's really hard to do that with a configuration.

Finally, you can also control the amount of tinkering a customer can do with CRM components in a solution package – and while these controls are not an overwhelmingly articulate instrument, you can choose whether customers can make modifications to the properties of some types of solution components.

Some examples of choosing configuration over writing code are using workflows instead of plug-ins, and business rules instead of JavaScript. Choosing configuration can potentially improve performance, require less maintenance, and decrease the amount of bugs in your solution.

Don't Copy On-Premises Design – use PaaS Instead

[4] Instructions for doing single sign-on between Dynamics CRM Online and your code deployed to Azure, as well as authorizing your code to call the Dynamics CRM Online web services for the logged-in user, can be found here: http://andrewbschultz.com/2014/11/26/how-to-authorize-an-azure-app-to-use-crm-online-web-services-without-asking-the-user-to-authenticate/

There are two main ways to deploy your code to Azure – using Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). Here are some quick heuristics to help you understand the difference:

IaaS

With Azure IaaS, you can put your code on a virtual machine (VM) in the cloud and run it just like it was on your own server. You turn on Microsoft Internet Information Services (IIS), create a virtual directory, set bindings, etc. – whatever steps you would take to configure your complementary web application in an on-premises environment. If you're looking to just dump some legacy code in the cloud, then IaaS will give you the environment most similar to the on-premises one from which it came.

However, you probably shouldn't just indiscriminately take an on-premises code base and put it in the cloud. There are better ways. If you have an Azure VM running web services, Windows services, or scheduled console tasks, you're not taking advantage of the cloud's unique architecture. These things can all be accomplished in PaaS.

PaaS

With Azure PaaS, more low-level management tasks are handled for you – you don't have to configure IIS, or patch the operating system on the machine. You just write your code and deploy it as a cloud service. In the background, web-facing code can go in a web role or an Azure Website, and non-web facing code can go in a worker role.[5] PaaS also includes other services available to your code,[6] such as storage as a service (like the SQL Database feature or a NoSQL database called Azure Table Storage), Azure Active Directory (which is Active Directory as a service), machine learning as a service, big data analysis as a service, scheduling as a service – you get the idea.

The caveat is that you need to develop your application in a certain way to run it in PaaS as a cloud service. You can't take a web application you built for an on-premises IIS server and deploy it whole cloth as an Azure cloud service. PaaS is the more sophisticated option, though – it scales more easily than IaaS (simply scale the web role up or out if your web site is getting more hits than expected, or scale the worker role up or out if your background processing tasks are more onerous than you expected). It also requires less maintenance.[7]

Design for the Azure SLA[8]

One of the great things about Azure is its service level agreement (SLA). Since you're not managing the hardware, you might wonder how you're going to make sure your service is stable by using good disaster recovery, high availability, and geo-replication deployment practices. To be subject to the Azure SLA, there are certain conditions you need to meet in the way you deploy your code and other assets to the service. For instance, you need to deploy your code to at least two instances of the web/worker roles or Iaas VMs, so if the hardware in the datacenters is serviced or a fault occurs in one datacenter, you still have a working instance. It's important to understand how to deploy your code to take advantage of the Azure SLA, and to meet goals for high availability, disaster recovery, and geo-replication.

Customizing for the Cloud

Unsupported customizations[9]? Bad idea. Not only are there fewer opportunities for you to make unsupported modifications in Microsoft Dynamics CRM Online, there is also a greater incentive to avoid making them. The purpose of Microsoft's guidance on supported and unsupported customizations is to allow your modifications to continue to work through new releases of Microsoft Dynamics CRM.

[5] Worker roles and web roles are two types of cloud services available in Azure. Under the covers, they are simply VMs that are created automatically for you and require no maintenance or interaction. The main difference is that web roles come with IIS enabled by default, although a worker role can have IIS turned on as well. They are monitored by Azure, and your application is maintained on these VMs by Azure. See http://azure.microsoft.com/en-us/documentation/articles/fundamentals-application-models/#CloudServices

[6] The services listed here in, like SQL Database and Azure Active Directory, can be accessed from code running in IaaS as well as PaaS.

[7] For a more in-depth comparison of PaaS and IaaS, see this blog post: http://blogs.msdn.com/b/hanuk/archive/2013/12/03/which-windows-azure-cloud-architecture-paas-or-iaas.aspx

[8] See http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-manage-availability/ and http://azure.microsoft.com/en-us/support/legal/sla/ to understand the terms and underlying principles of the Azure SLA.

[9] Supported extensions for Microsoft Dynamics CRM https://msdn.microsoft.com/library/gg328350.aspx

In Microsoft Dynamics CRM Online, these releases happen approximately every 6 months. The Microsoft Dynamics CRM platform could change in ways that break an unsupported customization, but supported customizations should continue to work through upgrades. While in an on-premises deployment, you may have been able to delay an upgrade indefinitely to avoid having to remediate your unsupported modifications, in Microsoft Dynamics CRM Online that's not an option. You should expect unsupported modifications to break at some point.

So what if you're deployed to Microsoft Dynamics CRM Online, and you want to test some customizations before deploying them? Microsoft Dynamics CRM Online allows you to add additional instances to a subscription, which can be designated as "sandbox" instances. In certain scenarios, sandbox instances may even be included with a Microsoft Dynamics CRM Online subscription. You can copy a production instance of Microsoft Dynamics CRM into a sandbox instance (including production data, if desired). You can also reset a sandbox instance, to wipe out any customizations you've made and prepare it for testing next time.

Staying Current

The release schedule of Microsoft Dynamics CRM Online is frequent[10]. This is good and bad – it's a rapid development pace with great new functionality being delivered frequently – but it's also necessary for ISVs and other solution builders to stay current with the release schedule. While supported customizations should not break with a new release, it's still possible that an update requires changes to your application. For example, the release of Microsoft Dynamics CRM 2013 introduced a new UI that was significantly different than that of the previous version. This required ISVs to make sure that their products made sense and were usable within the new UI paradigm.

It may be possible for individual customers to postpone the upgrade for a finite time period, but you shouldn't make your customers do so in order to give yourself more time to get ready. Any given customer could potentially have many ISV solutions, and the other solutions may be ready for the update to the CRM platform when it's released. You don't want to be the one holding the customer back.

Finally, there is also another reason to stick with supported customizations. With an accelerated release cadence, your time to prepare is short. It's much better spent bringing the new features and functionality into your solution than fixing unsupported code so it doesn't break a customer's deployment. This new functionality may include the ability to configure modifications that previously required custom code.

Part of your effort every to "get current" with each release may be the identification of areas where code is no longer needed to achieve the functionality you've built. This would have been possible in recent releases, including the 2014 Spring Wave and Fall Wave releases. If you have the discipline to proactively replace your code with configuration as it becomes available, you may see a general lessening of the maintenance burden for your application as Microsoft Dynamics CRM becomes more and more configurable.

Don't Jump to Customization

In the past, the approach of CRM solution builders has often been to write code as the first step to solving any problem. Today, the configuration capabilities of Microsoft Dynamics CRM provide more productive ways to solve problems. We should evaluate these configuration options before deciding to write code. Custom code is harder to maintain than configuration, harder to understand, more likely to break, and more likely to prevent upgrades of other system components ("yea, the custom code depends on that other thing. It's complicated code – we'd rather not mess with it, so that means we can't change this thing."). There are great advantages to being able to write custom code when it's necessary, but it can be a great weakness to do so unnecessarily. The CRM platform has been changing over the last several releases to allow solution builders to accomplish more with less custom code.

Avoiding too much complexity in your solution is always a good idea, but with the update cycle in Microsoft Dynamics CRM Online, it's critical. The less complex your solution is, the simpler your regression testing and maintenance will be for each release. Be kind to yourself, and don't over-code your solution.

[10] Get ready for the next release http://www.microsoft.com/en-us/dynamics/crm-customer-center/get-ready-for-the-next-release.aspx

UI Customizations

It's helpful to have a list of some of the customizations that people do to the UI in Microsoft Dynamics CRM on-premises that cannot be done in Microsoft Dynamics CRM Online, as well as the alternatives that may exist. Here is just such a list:

- **Custom ASP.NET pages:** if you had a button that would open up a custom ASP.NET page and retrieve or submit information from/to the Microsoft Dynamics CRM web service, you can't put that in Microsoft Dynamics CRM Online. There's no place for it. Are you sure you can't accomplish what your page is doing through the client-side scripting available with web resources? If you can, do it. If not, use the Azure approach mentioned under the heading "Use PaaS or IaaS in Azure When Necessary" in the "Cloud Principles" section of this paper, along with the instructions linked in an earlier footnote for configuring SSO between Microsoft Dynamics CRM Online and Azure.
- **Changes to the DOM in CRM:** Ill-advised in the on-premise solution, even more so in Microsoft Dynamics CRM Online, and unsupported in both types of deployments. If you need to do something crazy to the way fields on a form are presented, try embedding a web resource on the form that presents the fields in the way you desire. The web resource can retrieve and send the data for the fields from/to the CRM web service.[11] If your desired changes are to parts of CRM other than forms, sitemap areas, and toolbars, then you're out of luck.
- **Silverlight:** is only supported on Windows and Mac clients. So write your UI bling in HTML instead. With a cloud application, it's important to be cross-device compatible. You likely don't have an environment where IT mandates all the devices that can be used by the end-users.

Server-Side Code

- **Machine Resources:** you can't use the web.config in CRM code. You don't have access to the file system. You don't have access to a user's authentication credentials. Use the supported patterns for accomplishing these things in the cloud.
- **Plug-ins:**
  - If you're accustomed to deploying your plug-ins to disk in on-premises, you will need to deploy them to the sandbox[12] in Microsoft Dynamics CRM Online.
  - If you use 3rd-party libraries in your CRM code, you're probably accustomed to deploying those libraries to the global assembly cache (GAC) on the machine where your code runs. There's no GAC in Microsoft Dynamics CRM Online, and hence no way for you to authorize those third-party libraries. Use ILMerge to merge those libraries into your own assembly.[13]
- **Azure:** as mentioned elsewhere in this paper, long-running code and components like ASP.NET web applications that can't be deployed to Microsoft Dynamics CRM Online should utilize Azure. Also, this is true for any code that needs to execute with full trust.[14] Code running in Microsoft Dynamics CRM Online runs in the sandbox – if your code needs more, then you can deploy it to Azure and call it from Microsoft

[11] For an example, see http://blogs.msdn.com/b/devkeydet/archive/2012/02/11/displaying-a-lookup-as-a-dropdown-in-a-crm-2011-form.aspx

[12] For more about sandbox isolation, see http://msdn.microsoft.com/en-us/library/gg334752.aspx

[13] For more on ILMerge, see http://research.microsoft.com/en-us/people/mbarnett/ILMerge.aspx

[14] As the Microsoft Dynamics CRM 2015 SDK explains, "Microsoft Dynamics CRM 2015 and Microsoft Dynamics CRM Online support the execution of plug-ins and custom workflow activities in an isolated environment. In this isolated environment, also known as a sandbox, a plug-in or custom activity can make use of the full power of the Microsoft Dynamics CRM SDK to access the organization web service. Access to the file system, system event log, certain network protocols, registry, and more is prevented in the sandbox. However, sandbox plug-ins and custom activities do have access to external endpoints like the Microsoft Azure cloud service.

"Microsoft Dynamics CRM collects run-time statistics and monitors plug-ins and custom workflow activities that execute in the sandbox. If the sandbox worker process that hosts this custom code exceeds threshold CPU, memory, or handle limits or is otherwise unresponsive, that process will be killed by the platform. At that point any currently executing plug-in or custom workflow activity in that worker process will fail with exceptions. However, the next time that the plug-in or custom workflow activity is executed it will run normally. There is one worker process per organization so failures in one organization will not affect another organization.

"In summary, the sandbox is the recommended execution environment for plug-ins as it is more secure, supports run-time monitoring and statistics reporting, and is supported on all Microsoft Dynamics CRM deployments.

Dynamics CRM Online. Azure Service Bus may also be helpful here.[15]

Reports

- **FetchXML only:** When building reports for Microsoft Dynamics CRM Online, you need to write SSRS report queries in FetchXML. But once the query is written and the data is in your possession, you can build your report using the Business Intelligence Design Studio (BIDS) just like in on-premises. Upload the RDL to Microsoft Dynamics CRM Online just like in on-premises.[16]

- **Size & Scope of Data:** There is a limit to the amount of data it's technically feasible to store and report on in Microsoft Dynamics CRM Online. Or, your reporting needs may require mashing up CRM data with other data sources. In any case, there will be times when you want to use a reporting database that lives somewhere else – like a data warehouse running in Azure SQL Database, for instance. You can pull data out of Microsoft Dynamics CRM Online and report on it there. It's possible to move the data from CRM to Azure with standard ETL tools, such as SQL Server Integration Services and Scribe, which can run in an Azure IaaS VM.

- **Power BI:** While there are these limitations involved with reporting on data in the cloud, there are also benefits. Microsoft Power BI is a suite of tools that allows you to pull data from online and offline sources and easily analyze it, using the powerful capabilities of an online service.[17]

Custom Workflow Activities

You can deploy these to Microsoft Dynamics CRM Online just as you can to on-premises. They have to run in the sandbox though – no full-trust code.

Deploying in Hybrid Scenarios with On-Premises Components

Customers often have much of their infrastructure on-premises. They may be reluctant to use key applications in the cloud, because they don't understand the potential for hybrid scenarios. But there are many ways that Microsoft Dynamics CRM Online can operate as part of a hybrid IT portfolio. Using Microsoft Dynamics CRM Online doesn't mean the customer has to move its whole IT stack into the cloud.

Interoperability with Systems behind the Firewall

You have a lot of options when building interoperability between your code in Azure, Microsoft Dynamics CRM Online, and other applications. While any application that has the ability to call out to the internet can call the Microsoft Dynamics CRM Online web services, Azure also provides several tools that can structure or simplify the way you build interoperability.

- Azure Service Bus enables your code in Azure to connect to other devices or services that would otherwise be difficult or impossible to reach and communicate with.

- Azure Virtual Network allows you to create a network where IaaS VMs and PaaS services such as web roles can communicate with each other and with firewalled systems as if they were on the same network.

- Azure BizTalk Services is a cloud-based service that allows you to pass data back and forth between applications in different locations.

It's possible to set up a VM in Azure IaaS and run any data transfer application you would normally install locally.

In addition, Microsoft Dynamics CRM Online only supports execution of custom workflow activities if they are registered in the sandbox … Sandboxed plug-ins and custom workflow activities can access the network through the HTTP and HTTPS protocols." For more information, see the SDK topic "Plug-in isolation, trusts, and statistics". http://msdn.microsoft.com/en-us/library/gg334752.aspx

[15] For more on Azure Service Bus, see http://azure.microsoft.com/en-us/documentation/articles/fundamentals-service-bus-hybrid-solutions/

[16] For a helpful resource on FetchXML reports, see http://blogs.msdn.com/b/devkeydet/archive/2012/06/04/converting-a-sql-report-to-fecthxml.aspx

[17] Power BI will not be available for the Office 365 Government Community Cloud initially, which means that customers who want their entire footprint of applications to be confined to the Government Community Cloud will not be able to use the Power BI service. However, many of the capabilities grouped under "Power BI" are actually features of Microsoft Excel, such as Power Query, which allows you to query Dynamics CRM Online data to build advanced analytics and mashups with data from other sources in an Excel spreadsheet. Excel spreadsheets can be uploaded as reports in Dynamics CRM Online.

Email Router

You can deploy the email router to Azure IaaS if you need to. However, server-side synchronization is a better alternative that simplifies CRM and email interoperability by allowing you to configure it without installing the email router component.

Being Aware of How Your Customer will Administer your Solution

Storage

Customers only get 5 GB when they sign up for Microsoft Dynamics CRM Online. This number goes up based on the number of user licenses the instance consumes, and extra storage can also be purchased for a reasonable monthly price. When you're building Microsoft Dynamics CRM Online solutions, however, it's crucial to remember that your customer will pay for extra storage. Consider saving things like documents or other storage-intensive objects elsewhere, where storage is cheaper (SharePoint Online and Azure storage are two good candidates).

As a solution developer, there are tools you can use to estimate the storage you will have available to you before you start building, so you can design your solution to work with the storage available to it. The SureStep materials[18] include a sizing worksheet that can help you estimate the storage a customer solution will require. Also, make sure you understand that storage for a tenant (i.e. a single customer subscription to Microsoft Dynamics CRM Online) is shared across instances of Microsoft Dynamics CRM Online. In other words, if you have a production instance, and several non-production instances, or more than one production instance running under the same Microsoft Dynamics CRM Online subscription, the storage available to your tenant has to supply all of those instances.

It's important to stay on top of this, because it's possible to run into situations where a lack of storage will block you from doing what you need to do. For example, if you want to copy your production instance into your non-production instance to test some customizations you're developing, but you don't have enough open storage space for another copy of your production instance, you'll be unable to perform the copy.

Database Indexes

As mentioned elsewhere, it's possible to set indexes on the database, but it has to be accomplished by Microsoft Dynamics CRM Online service engineering through a support request.

How Authentication is Different in the Cloud

Claims-based Authentication

When you build solutions for Microsoft Dynamics CRM Online, you have to use claims-based authentication. Know this. But you should also consider the benefits of having single-sign-on capabilities with Office 365 and apps you've deployed to Azure. This may be something that makes your solution much more useful.[19]

Microsoft Dynamics CRM Online uses Azure Active Directory as its Identity Provider. Azure Active Directory is administered by the Azure customer, and the customer owns the relationship with the users in Azure Active Directory, just like it would with on-premises Active Directory. One of the benefits of having Azure Active Directory as your cloud identity provider is that it allows you to tie in to on-premises identity providers (henceforward "on-prem IDPs") if you so desire – Active Directory most easily, but also others. There are different levels of interoperability possible between Azure Active Directory and an on-prem IDP, some of which are discussed briefly in the sections below.

Cloud Identity

You have the option to house your identity

[18] for more info on SureStep, see https://mbs2.microsoft.com/Surestep/default.aspx

[19] See instructions for having your Azure cloud application perform SSO with Dynamics CRM Online, and authorizing your application to call the web service through the authenticated user, here: http://andrewbschultz.com/2014/11/26/how-to-authorize-an-azure-app-to-use-crm-online-web-services-without-asking-the-user-to-authenticate/

completely in Azure Active Directory. Accounts are created and managed in the cloud, with no awareness of on-prem IDPs. This, of course, means that if the on-prem IDP exists, a user will have a different account for the applications served by Azure Active Directory and those served by the on-prem IDP. This option is most typically chosen when no on-prem IDP exists, where the on-prem IDP is of such a type that no interoperability is possible, or when a business application is being deployed to the cloud quickly and creating interoperability with an existing IDP would cost too much time or money.

Synchronized Identity

In the synchronized identity scenario, the user's identity lives in the on-premises Active Directory (or other compatible identity provider). The user's directory information is then synchronized to Azure Active Directory, so the user can log into the cloud applications served by Azure Active Directory using his/her corporate credentials.

There have been several different tools for synchronizing identity between Active Directory hosted on-premises and Azure Active Directory. At the time of this writing, some methods are deprecated or will be soon. Some of them only sync in one direction (on-premises to cloud), which means no changes to the directory can be made in Azure Active Directory.

Others have features, such as password writeback, that allow certain changes to be made in Azure Active Directory and synced back to the on-premises Active Directory. Make sure you understand the nuances of the synchronization tool you use if you're going to synchronize identity.

When user sign-in occurs with a synchronized identity, it's handled in Azure Active Directory, with no communication down to the on-premises Active Directory deployment. This setup is sufficient for many scenarios.

Federated Identity

Federated Identity further integrates the on-prem IDP with Azure Active Directory. In addition to synchronizing identities to the cloud, the sign-in operation is federated, which means that the login process is redirected to the on-premises IDP. In this scenario, this on-premises IDP can be Active Directory Federation Services (ADFS) or other compatible providers.[20]

With this federation, a user can log in to Microsoft Dynamics CRM Online using their corporate credentials. As Ross Adams, Paul Andrew, and Aanchal Saxena explained at TechEd North America 2014,[21] Federated Identity is good for scenarios like the following:

- There's an existing ADFS deployment
- There's already a third-party identity provider in use
- There are multiple forests in an on-prem Active Directory deployment
- Smart Card or Multi-factor Authentication is in use on-premises
- Hybrid applications or hybrid search is required
- A web accessible password reset is required
- You require sign-in audit and/or immediate disable
- Client sign-in restrictions by location or work hours is required
- There is a policy against synchronizing password hashes to Azure Active Directory

Conclusion

Creating solutions for the cloud is different. It requires thinking about the system architecture in a different way. But making this leap is an imperative for almost anybody selling business or government solutions today. Keeping up with the market requires us to make the necessary adjustments, and allows us to embrace the attendant benefits of building solutions for the cloud.

[20] See the "Works with Office 365" program details: http://aka.ms/ssoproviders

[21] http://channel9.msdn.com/Events/TechEd/NorthAmerica/2014/OFC-B317#fbid=?hashlink=fbid