**Sandbox Solutions vs Farm Solutions**

Now, as I mentioned there are two kinds of .wsps: Sandbox and Farm. Well, one thing I want to mention to you is that the .wsp itself doesn't have a flag in it that looks like IsFarmSolution is equal to true. In other words, if I was to give you a .wsp and I say, "Hey, is this a farm solution or sandbox solution?" the answer is you can't really tell. Sandbox solutions are more restrictive in what they can do. And what differentiates a solution from a sandbox to a farm is how you deploy it, right, and how you run it. So a sandbox solution can be uploaded directly through the browser; a farm solution, on the other hand, has to be installed using PowerShell commands. A sandbox solution can also be installed using PowerShell commands, but a farm solution cannot be uploaded from the browser-based UI for instance, right? So depending upon how you deploy a solution -- it's going to run as a farm solution or sandbox solution -- the golden rule is that every sandbox solution will run as a farm solution, but not every farm solution will run as a sandbox solution. Now, you may be wondering, you know, that, "Maybe I just should just write farms solutions because everything works in farm solutions." The answer is a big no. Right? You should avoid writing farm solutions. I'll get to that in a second. So the first question is: What can I write as solutions? And the answer is anything SharePoint 3. And this is a small incomplete list of what can be considered SharePoint 3. All of those can be written as solutions, or content types, site columns. These are various SharePoint concepts. And you know, I'm going to demonstrate this using Webparts, but remember, that's just one possible thing that you can write as solutions. There are many other things that you can write as solutions. A farm solution, which typically you would copy onto the C drive of the server -- RDP into the server and use PowerShell commands to install it -- a farm solution has the ability to make changes to the file system and also make changes to the content database, right? A sandbox solution, on the other hand, will never, ever touch the file system. Now, if you were to take the .wsp file of a sandbox solution -- something that is running as a sandbox solution that can run as a sandbox solution like a Webpart for instance -- a Webpart, the logic of it is implemented in a .dll. And then you might think that, you know, the .dll has to go in the GAC because if you were to open the .wsp and read the structure of it, literally says that this goes in the GAC. But a Webpart can work as a sandbox solution. So is the sandbox solution putting things in the GAC? And the answer is "absolutely not."

### [**The Sandbox Solution Code Service**](https://app.pluralsight.com/player?course=understanding-sharepoint2013-development&author=sahil-malik&name=understanding-sharepoint2013-development-m3&clip=9&mode=live)

For sandbox solutions to work what you have to do -- you go to this manage services on server -- and you have to ensure that this service is running: Microsoft SharePoint Foundation sandbox code service. I'm going to leave this dot for now because I want to demonstrate that it causes an error if you try to run sandbox code with this running, right -- with this stopped. So I'll start this in a moment. So before we dive into actual code,

**let's do a quick comparison of sandbox solutions and farm solutions.**

* So sandbox solutions are easier to deploy. A site collection admin or somebody with full control can deploy them. How do you deploy them? You go to your solution gallery and you upload it, right? It's as simple as that.
* Sandbox solutions are also better monitored. These are those monitoring metrics that I showed you in the slide. They can be monitored both by a site collection admin and a farm admin.
* They are restricted in what they can do, as in that's the KAS policy and the trust level, etc. that they've used to restrict sandbox solutions.
* And they can do limited damage, as in sandbox code, the maximum damage it can do is that it can bring all sandbox solutions on your site collection down, right? So it will never hurt the farm, it will never hurt other site collections, but it can hurt your site collection.

**Managing Sandbox Solutions in Office365**

In SharePoint Online in Office 365 when you log onto Office 365 -- this is the 2013 version -- you'll see here that they have given you certain storage quotas and server resource quota that, as an administrator of your SharePoint Online environment, they've given you certain amount of the points and you can distribute them across your site collection. So perhaps the limit per site collection in SharePoint Online is a little bit lower, but this is how they've solved it in SharePoint Online. The number of points -- you as an administrator can decide who gets how many points. Now, when I talk about apps you will see that this monitoring on sandbox solutions is both a good and a bad thing because once you hit this limit, you know, your code stops working. So apps are not monitored in this manner. So while that's not great that they're not monitored, but the reality is that you can do a lot more than apps.

**This is one of the biggest problems with sandbox solutions, is that they're almost too restrictive, right? In comparison,**

**farm solutions are a pain to deploy: You have to hunt down your farm administrator;**

you have to have somebody who's got admin rights to your SharePoint environment; can RDP to that server, etc. So deploying farm solutions is more difficult. There is no monitoring support for them. You can write something yourself, but they're still difficult to monitor. There are no restrictions on them, and they can do lots of damage, right? Basically they are under the unrestricted trust level on your SharePoint server. So the theory here is that avoid writing farm solutions. But we can't one hundred percent avoid it, so at least try and minimize writing farm solutions if you can.

**Farm Solution is Pain**

**You will find farm solution:**

**Central Admin > System Setting > Manage Farm Solution**

So again, farm solutions are a pain to deploy. I deployed them through Visual Studio, but what I would encourage you to do is that fire open PowerShell, right, and look at:

How would you deploy a farm solution using, you know, by hand? So how do you do that? Basically say help SPSolution. And you will find commands here that look like Add-SPSolution, Remove-SPSolution, Update- and Remove-SPSolutionDeploymentLock. So even if you look at the first three -- add, remove, update -- and the things that allow you to deploy these as well. So you know, basically what happens is that if you go into system settings and you click on manage farm solutions, the add solution will add it here. Visual Studio's done this for me, but you can also do this through PowerShell. And then you can, you know, Install-SPSolution and so on and so forth to be able to get it out to certain web applications or attract it from there, Uninstall-SPSolution, and so on and so forth. So you have to have access to the server, you have to be able to copy the .wsp file onto the C drive of the server to be able to deploy a solution and add a solution, and so on and so forth. Right?

So you see here that farm solutions are a pain to deploy. There is no monitoring built in, there are no restrictions, and they can do tons of damage, right? I'll show you something interesting before I totally close this video. You see, there is an object here called as SPContentDatabase. Right? Content database. And this is actually available in SharePoint administration, right? Or actually, there's another one called SPApplicationPool. Right? So appPool.Username gives you, you know, the username of the user ID running this application pool. What do you think this property gives you? Password. Right? Now it's deprecated. In SharePoint 2010 it was deprecated, but in SharePoint 2007 this would have just given you the password of the Windows account that is running the application pool. Right? So you see my point that this is really, really -- can be really harmful. Let me also show you a couple of other things. You can say SPSecurity.RunWithElevatedPrivileges. And this runs code as a special account called SharePoint/system, which is access with regard like admin level access to the SharePoint farm. You don't need a password even for that. Using SPSite, site is equal to SPSite. And here also you can choose to pass in a userToken. And the userToken is just a property on SP user object that you can easily find out. So you can impersonate as any other user on a SharePoint farm solution, right? These are things that sandbox solutions don't allow, but farm solutions do allow. In some scenarios you need these things. But remember they're a lot of power and you have to be conscious of, you know, what you're doing here. So again, avoid writing farm solutions if you can, but you can't one hundred percent avoid them, so try and minimize them. So a quick summarization. What have we looked at so far? We've looked at, you know, an introduction to PowerShell; console application; SharePoint object model; client site object model. And we took a brief overview of sandbox solutions and farm solutions. Now, these are just overview level topics so far. I'm just giving you a taste test in this course of what is possible to write in SharePoint. The last part that I'm going to talk about within this course is writing SharePoint apps. So I'll catch you in that video. Thank you for watching. My name is Sahil Malik. I'll see you in the next video.

**The Need for Apps**

The solution packages that you looked at is, a fabulous way to add functionality into SharePoint as much as we try and award developers, we know that developers are, you know, a necessary evil. The problem is that the tools that the developers have had so far until SharePoint 2010 to be able to deliver functionality for SharePoint or sandbox solutions and farm solutions. One of them is too restrictive, so people ended up not using it as much and the other is too dangerous. Right? So wouldn't it be nice if it had something in the middle? And that something in the middle is apps. Apps are much more manageable than farm solutions. They will never ever harm the SharePoint server. They'll never harm the SharePoint server is true, but there are still security considerations that you need to consider about apps, and I'll be talking about those in detail when I talk about apps, right? But one thing that you can be absolutely sure of with apps is that they will never ever damage your SharePoint server, right? But they have the ability to do things that sandbox solutions weren't able to do.

**Apps Philosophy and Design Architecture**

So the golden rule about apps is that you will never ever be able to run any code on the SharePoint server. So anything that ends in the extension.dll cannot run on the SharePoint server and be a part of an app. Right? It has to be on a separate server. So SharePoint apps don't live on the SharePoint server, they can be surfaced out of a SharePoint server. They can be installed on a SharePoint server but they never execute on the SharePoint server. So they either execute on a separate server outside of SharePoint, or they can run inside of the browser using JavaScript, or Silverlight, or something like that. And this external to SharePoint code generally will talk to SharePoint using Seesaw, or REST API, or something like that, and they would authenticate themselves and get permissions using open standards like OAuth.

**AppWeb vs HostWeb**

So speaking of SharePoint-hosted area, the way this works is that a user finds an app, a user installs an app, and the process of installing an app will provision a new SPWeb, a new site; and this site is a child site of where you installed the app. So where you install the app is called as the "Host Web," and the SPWeb that got created for you with the installation of the app is called as the "AppWeb." There's a big advantage of doing it this way that, you know, when you uninstall the app, you just delete the AppWeb and everything that got created with the AppWeb is removed, right? So it's a great way for us to clean up nicely when the app is removed.

**The App Package**

So, remember there are solution package, you will deployed it as a .wsp, which was a .cab file. Deploying an app is a clear zip file which is based on the same zip file format as docx and xlsx as in pptx files; and they contain a file like AppManifest.xml. Just like the .cab or .wsp had a manifest.xml, the .app file has got an AppManifest.xml. Then the AppManifest.xml tells SharePoint what is contained inside of this app, right? And the app may have, you know, a number of things, but optionally the app may also have a wsp inside of it. You may want to know why does an app have a wsp inside of it because that AppWeb that gets provisioned, right, this AppWeb, maybe you want to create a list inside of it, right? Maybe you want to provision a BCS external list, for instance, or some SharePoint artifact that you might find useful for the functionality of your app. And that -- creating that would be the responsibility of that contained wsp. So the app can contain the appmanifest.xml which points to like the start page location or a bunch of other things, but the app itself cannot contain any dlls, et cetera, so maybe they're on a separate provider hosted area, et cetera. But in addition, the app can optionally contain a wsp inside of it. Right? This will all become very clear when you see this running in the demo. So again, as I talked about AppWeb or HostWeb, this is a very important concept that we need to know about and understand very thoroughly. HostWeb is where you install the app, the process of installing the app created, a subsite to the HostWeb, and that subsite is called as the AppWeb.