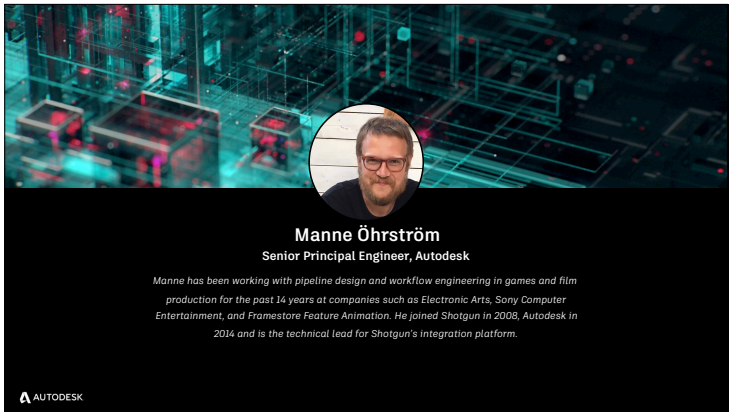
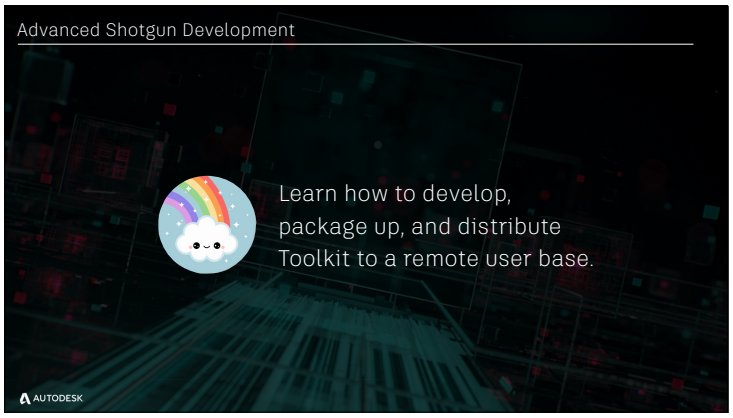
Welcome to this talk about advanced shotgun development!



Let me start by introducing myself!

My name is Manne. I have been designing pipelines and tools for vfx, feature animation and games companies for the past 15 years.

I initiated the Toolkit project about 5 years ago.

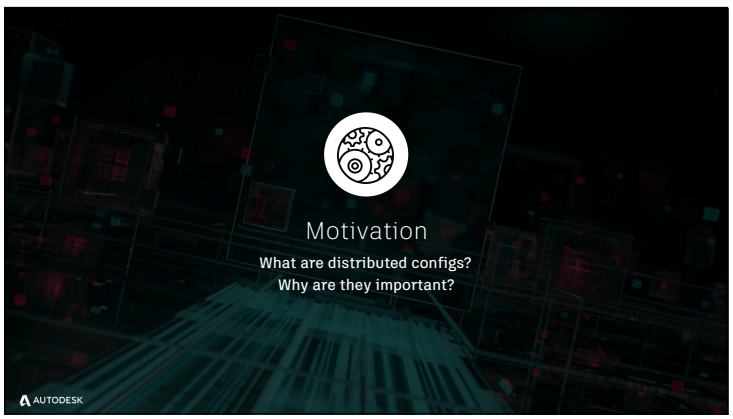


So for the next 30 minutes i will be talking about remote deployment and what we call distributed configs.

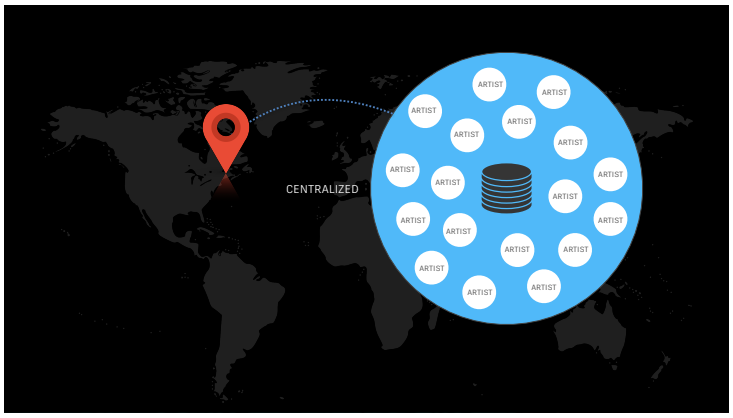
Just curious before we dive in:

- how many of you have used the toolkit zero config out-of-box integrations?
- how many of you have set up a toolkit project?
- How many have used the bootstrap API, zipped configs and remote deployment APIs?

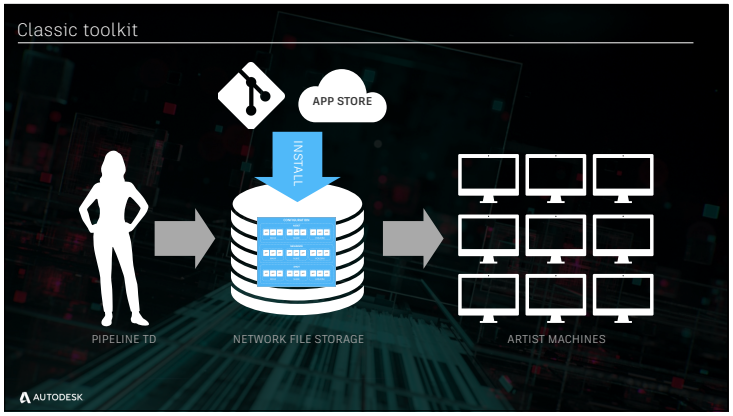
So in this talk we will be talking a lot about how to setup toolkit - and while its not required, it definitely helps if you already know what a more traditional toolkit setup looks like!



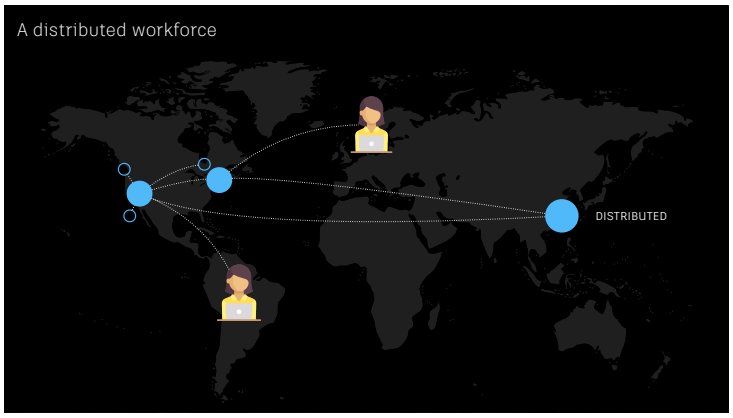
So this session is all about distributed configs.
Before we start, let me talk a bit about the background.
What are distributed configs and why are they important?



Traditionally, a lot of studio setups were in a single geographical location. All artists would all access a large shared network storage where files and data would be kept
This is what toolkit was originally designed for and what we today call a **centralised** configuration.



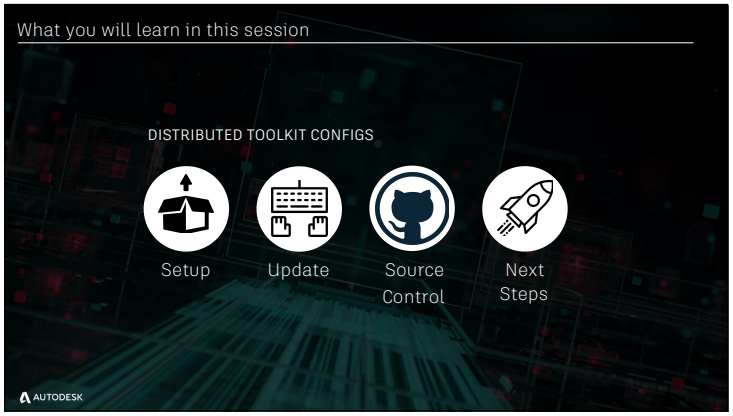
- The setup would look something like this.
- Using the SG desktop setup wizard, a TD would select a config and a choose a folder on the shared storage where to put it.
- The setup wizard would make sure all of the required tools would be downloaded
- and be made available to all the artists in that location



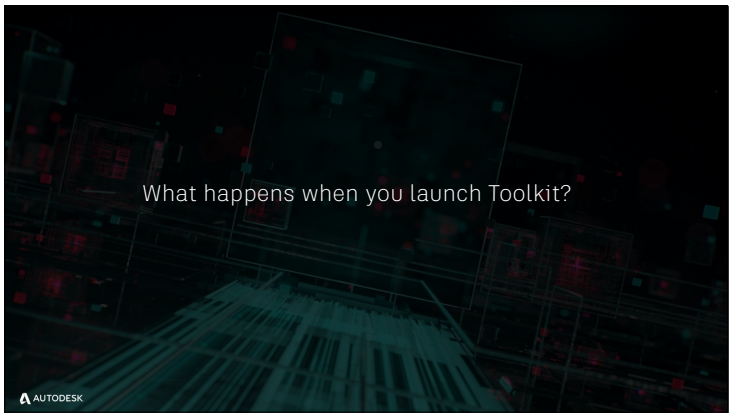
Today, VFX studios are more complex and often spawn multiple geographical locations.
In addition to that, it is common to have remote workers and outsource vendors you want to exchange data with.

This is where Toolkit distributed configs come into play.
Distributed configs don't need to be installed in a central storage.
Instead they are downloaded from Shotgun and installed on the fly.
This means that anyone who can access the shotgun site can also access the toolkit pipeline.

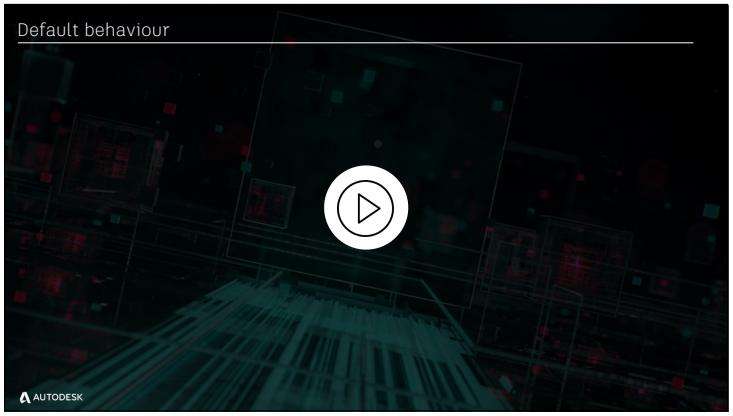
Distributed configurations are useful for many reasons, and can be used in a wide variety of ways.



This session is all about these distributed workflows.
I'll show you a bunch of hands on examples how you can work with these setups.
I will show you how to set up a distributed config
How to make updates and changes to it and how to push those changes out to production.
We'll talk about how and why source control is a great fit with distributed configs
Lastly, i'll touch on what lies beyond the first setup

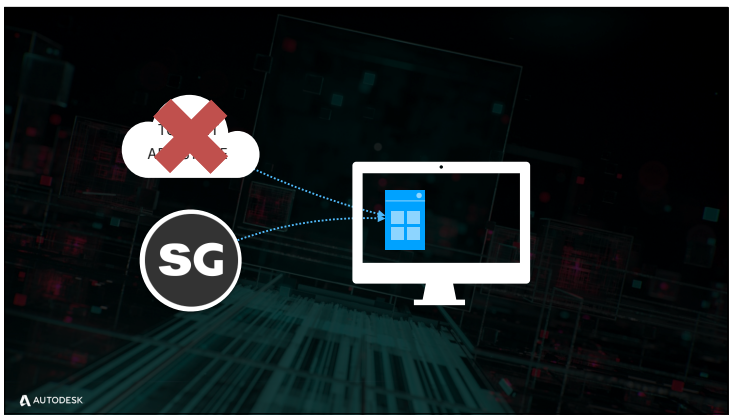


To kick this off, let’s take a quick look at what happens with
Shotgun Desktop and toolkit if you don’t configure it at all.



Let’s launch shotgun desktop and log in to our site.
I see a list of projects.
Clicking into a project, i get presented with a list of launchers and toolkit apps.
From here i can launch Maya, Nuke and other application and they will have toolkit apps available too.

So what happened there?
Behind the scenes, when Shotgun desktop starts up, it has built in logic to connect to the toolkit app store and download the latest basic configuration (tk-config-basic) directly into a cache folder on the user's machine. Once it's downloaded it, it downloads and caches all the apps and engines defined in the config automatically and starts things up. No project setup is needed, all this happens on the fly and works for any user anywhere.



This default behaviour that is built into Toolkit is the simplest example of how a distributed configuration works.

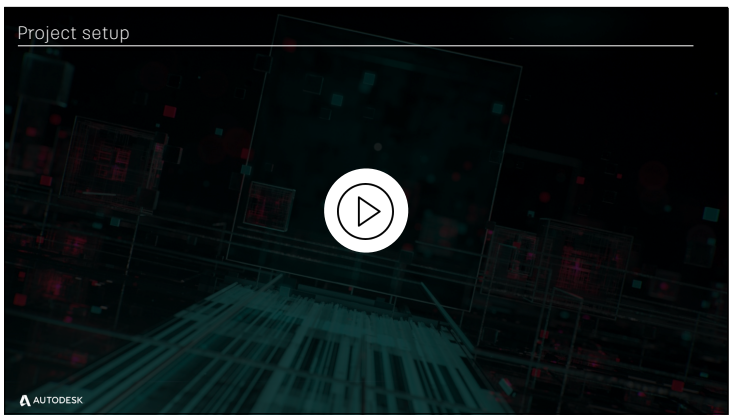
A configuration and all its dependencies are downloaded from the cloud to a user's machine - on the fly!

So we were thinking - hmmmmm - what if we could utilise the same logic for the pipeline configuration for a project?

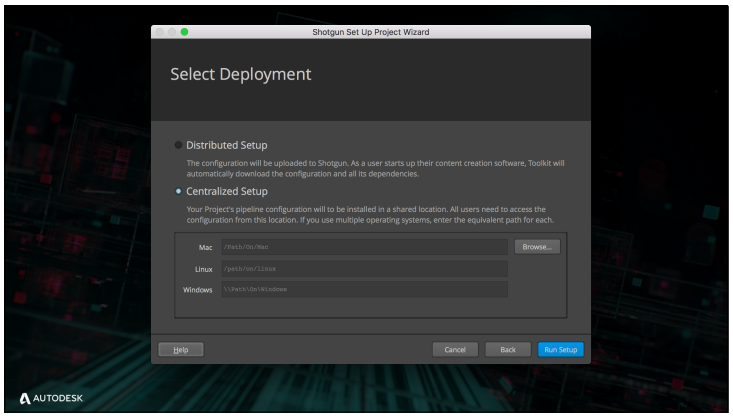
So Instead of having the Shotgun Desktop download a default configuration automatically
from our Toolkit App Store, we could set things up so that our artists are pulling down a config our our choosing from our own Shotgun site..

Let’s take a look at how a distributed Shotgun project can be set up. This is all done via the project setup wizard.

Just a disclaimer that we are in the process of adding a bunch of new, exciting features around distributed configs. I am about to show you some brand new things that we are working on that will be released very soon.



Let’s set up our project using the Toolkit project wizard.
We go in via the menu and go through the usual setup steps.

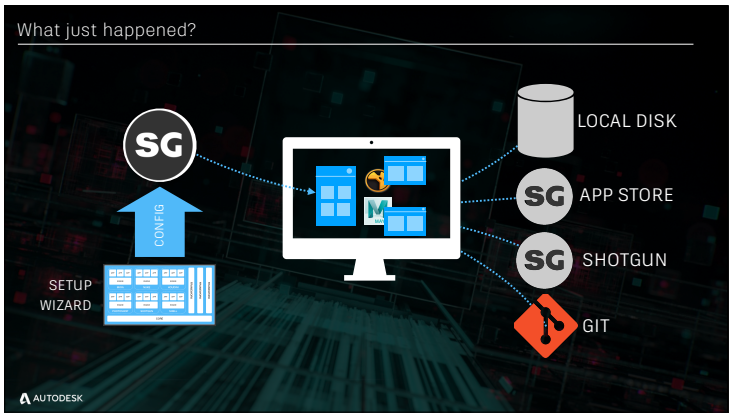


We have updated the UI here to handle distributed setups.

If you go with a centralised setup, this is the classic way to set things up - it gets installed in a central location, you get a tank command etc.

If you select distributed setup, the configuration will instead be uploaded to Shotgun.

We have updated the UI here to handle distributed setups.
If you go with a centralised setup, this is the classic way to set things up with a tank command



So what just happened?
The config wizard uploaded a zipped up config to our Shotgun Site.
The artist launches the sg desktop and logs in. At startup, the desktop application downloads the configuration from Shotgun
The configuration is analysed and any dependencies that don't already exist locally are downloaded.
Via the desktop application, content creation applications can be launched and the pipeline tools will automatically appear to the user.

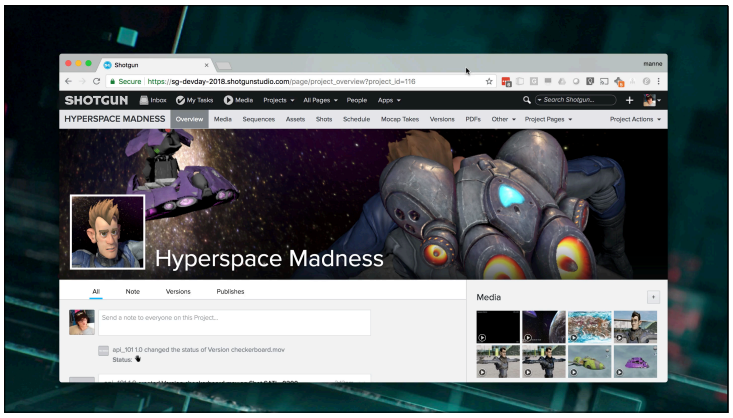
This pattern means that users anywhere can get the pipeline tools delivered from Shotgun.
If a user can log into Shotgun, they are also able to get the pipeline tools delivered directly to their machine.
If the config in Shotgun changes, the machines will detect this at startup and update.

Please note however, that distributed configs are not handling remote file transfers. What we are talking about here is just distribution of toolkit apps, so pipeline tools.

If you have remote artists, you will have to transfer files separately. This can be scripted by reading for example published files out of Shotgun and using that to drive file transfers.

Like you saw in the example earlier, a distributed config can have templates and storage.

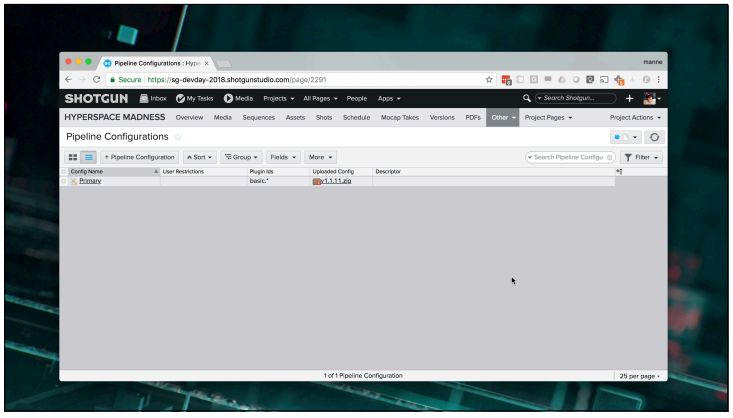
Another benefit is if your shared storage is optimised for large files, especially if it's a windows storage, it's may not be great to install toolkit on. In this case the distributed configs can significantly increase the performance of your setup.



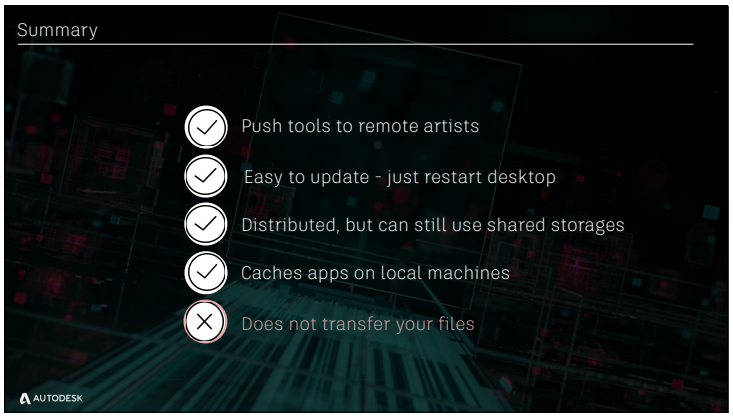
So let's take a look at what the setup wizard created in Shotgun.

What did it actually do?
Let's look at our project and its pipeline configurations.
So actually, it just automated some very simple steps that can easily be done manually or by a script.
It created a primary pipeline configuration for the project and upload a zip file to an uploaded_config field.

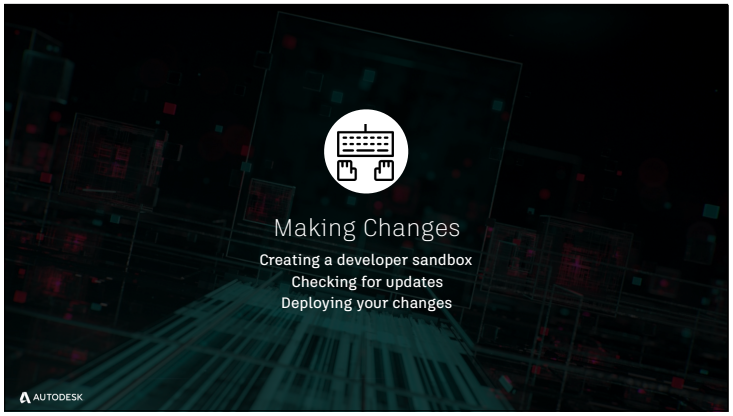
On a side note, this uploaded config field will be enabled by default in SG v7.11



There is nothing magical here, we can download this zip file and if we unpack it, we see that it just contains a standard toolkit config.

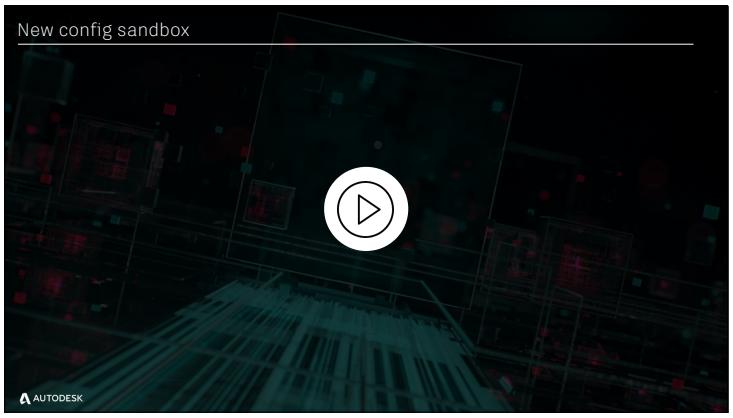


That's all it takes to set up your distributed config. So to summarize:
This workflows make it easy to push tools to any artist anywhere
Pushing updates is a matter of uploading a new zip to shotgun. artists get it when they restart their tools.
You can still use shotgun's local storages and template system
All apps, engines and frameworks needed to run are automatically cached on the user's machine
Please note that this system does not transfer your files, not yet anyways :)

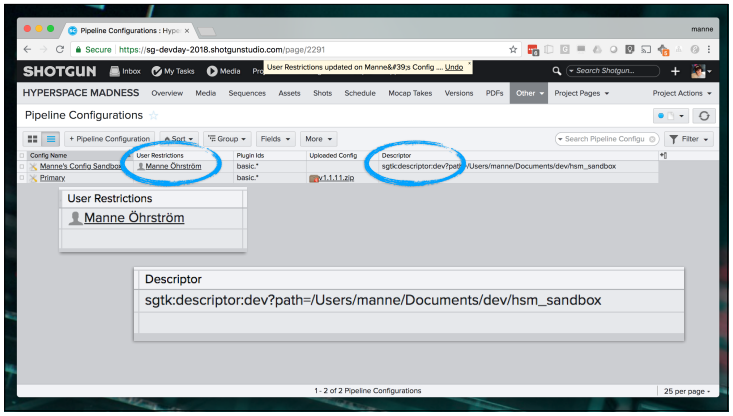


So having a zipped up config is great. But it's kind of locked down.

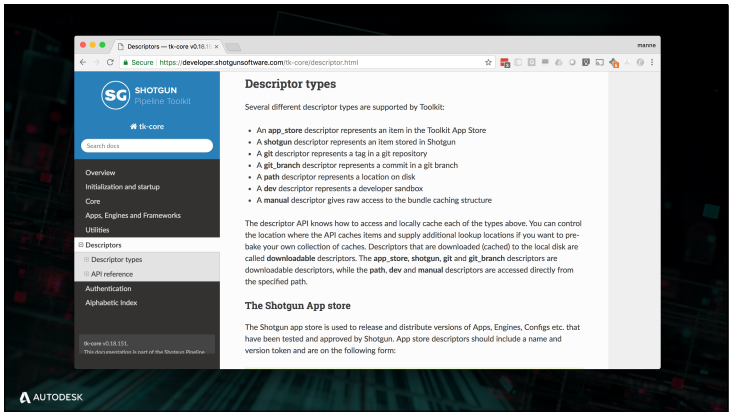
But how do we edit and tweak it interactively?
How do we dev?
Where does it live on disk if it's downloaded every time on the fly?
Where is the tank command?
In this section i'll show a typical change cycle.



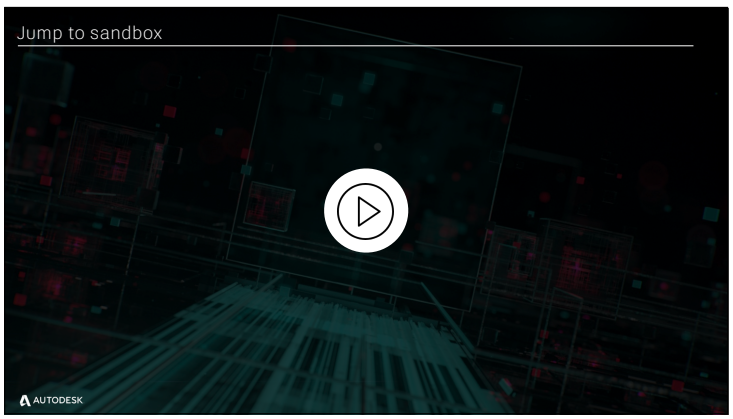
All these new distributed setups will come with some new tools.
On the context menu, you can now choose to create a ‘new config sandbox’
Similar to the clone configuration for centralised configurations, there is a convenience app in Shotgun
desktop that sets up a configuration sandbox for you in a couple of simple steps so you can begin
making config tweaks, hook changes or app development.



Let’s flip back to Shotgun again to see what has happened there.
A new Pipeline configuration has been created. Two things are worth noting:
It is restricted to my user only, meaning that no-one else will see it.
Instead of using an uploaded config, it uses what we call a descriptor URI, pointing at a local disk location.
This URI is a flexible system and Toolkit API to describe remote locations.

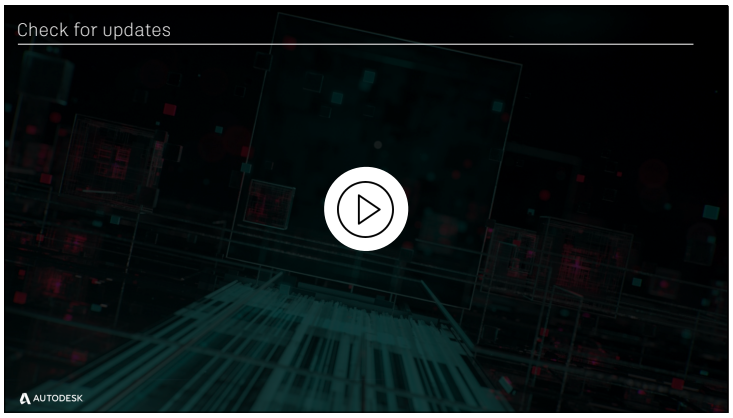


A range of descriptor types exist and you can use these the orchestrate more complex configuration distribution scenarios, for example using git. But that goes beyond the scope of this talk.

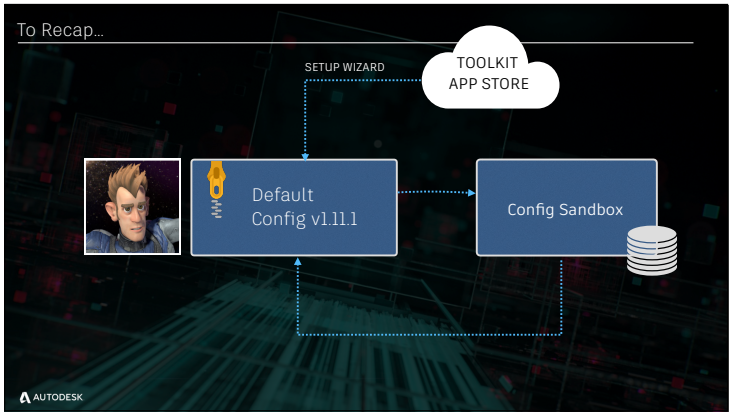


Once we are inside our config sandbox, we have a couple of more options.

I can easily jump directly from Shotgun desktop into my dev area and start tweaking. To reloading your changes just hit the reload and restart button.

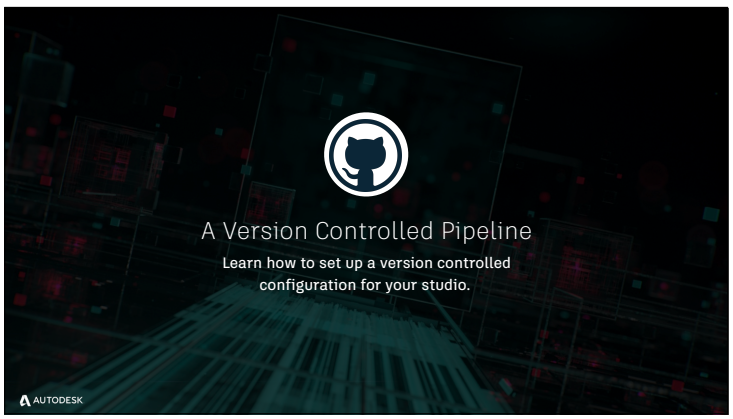


Another common task is to check for app and engine updates. For distributed configs, there is an app that wraps round the tank command update command for easy access.

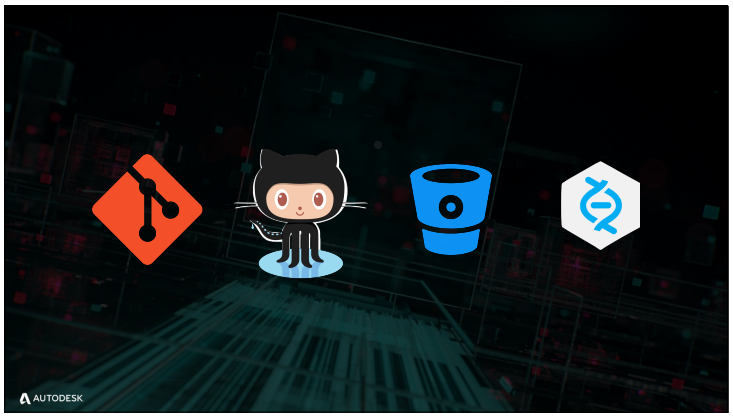


So to recap, we started with our Shotgun project
We used the setup wizard to grab the default config from the toolkit app store and uploaded it to Shotgun
We then created a configuration sandbox on our local disk and copied the configuration across so we can edit it.
Next step is to push our changes back into production

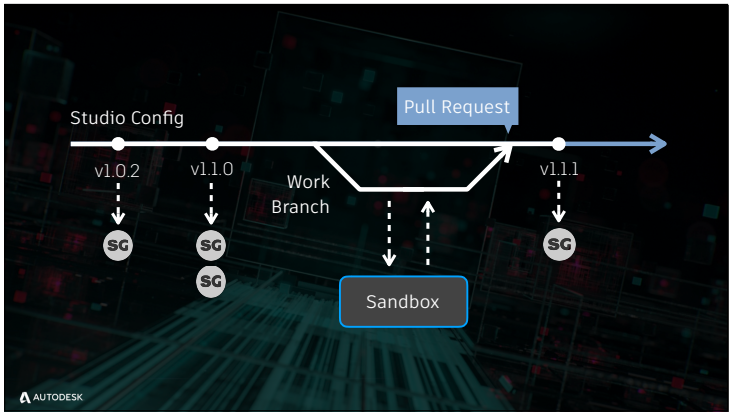
This is super easy!
It's just a case of zipping up your configuration and uploading it to Shotgun.



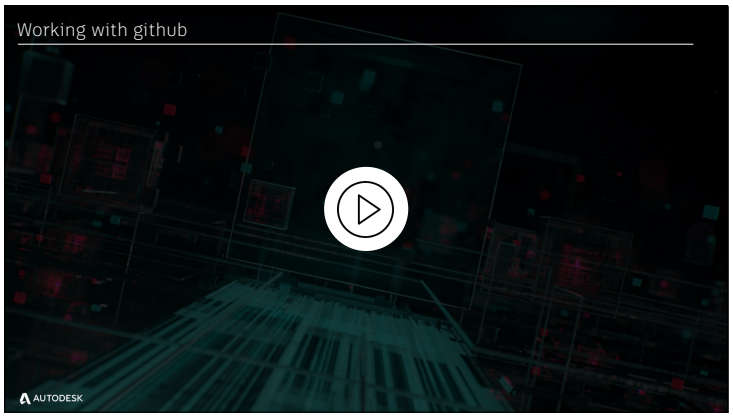
However, these workflows tend to be hard to maintain.
Sometimes things go wrong and you need to figure out what has changed and why.
This section of my talk is about how you can easily use source control in conjunction with these workflows.



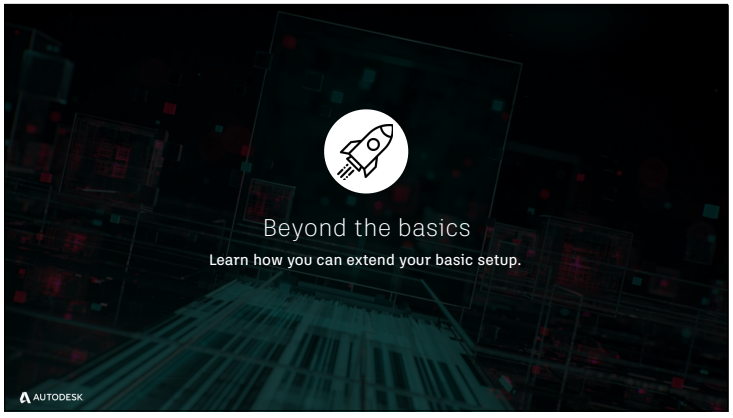
You can use any SCM you want. We'll be using github for our talk.



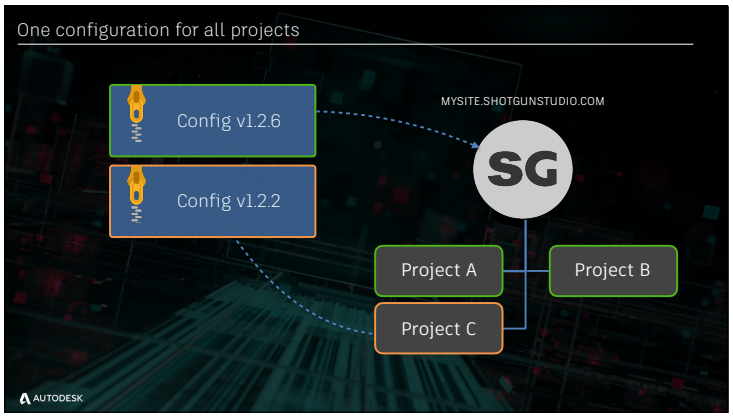
There is a studio configuration git repo
Tags are created and for each tag, a zip file is uploaded to Shotgun
Sometimes you update all projects, sometimes only some
When you want to do work, you create a branch
You can then set up an empty config sandbox for your shotgun project and check out your branch in there.
Once you have done you changes and tested them, push them back to git and create pull request for review.
Once it's been merged in, you can tag up a new release and deploy.



with something like github, these standard workflows makes for some great tracking of changes.
you can easily see, review, discuss and diff each PR
You can easily see the list of changes of the master branch
And you can manage your tags and releases and github creates zip files for them that can be uploaded directly into Shotgun.

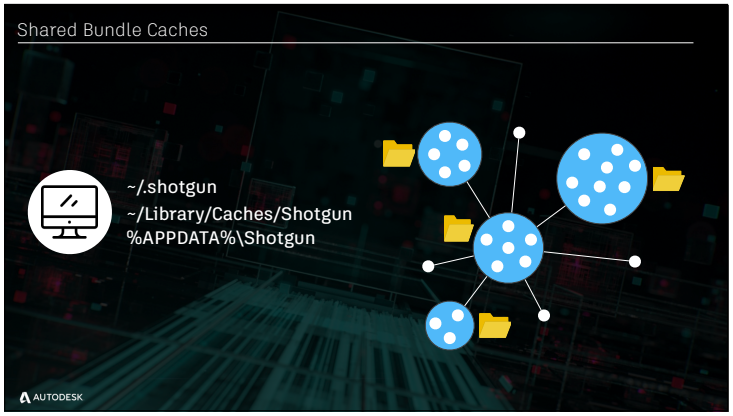


So now we have shown the basic tools and dev cycle around distributed configs.
Now we are going to dive into a bunch of specific features and patterns that we have added over the past year as we have been trying to help people get up and running with these kinds of setups.



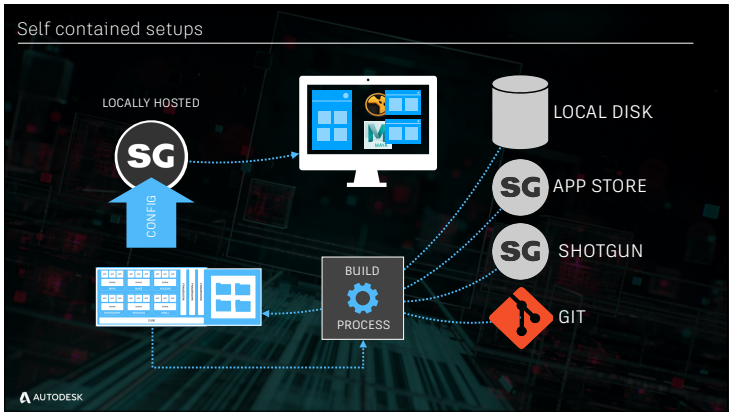
One cool thing that is possible to do with distributed configs that are not possible with classic configs is to set up a site level.

If you create a pipeline configuration in Shotgun and don't assign a project, it becomes a site-wide configuration.
Every project you create will then automatically pick this up. No need to use the project setup wizard!
If you do have a project which requires a special config, you can use the project setup wizard and a project specific config gets created which takes precedence.

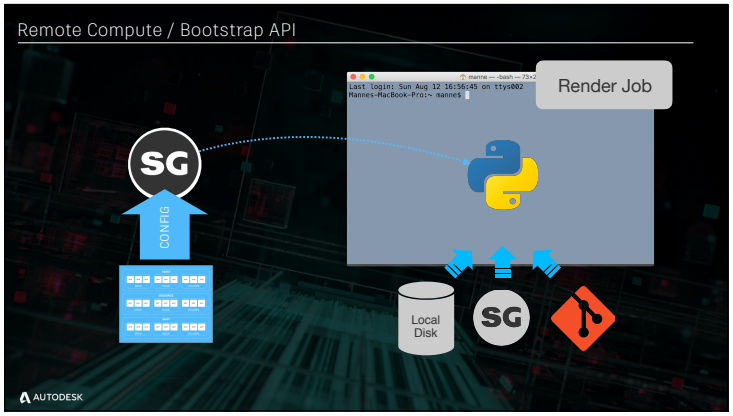


When Toolkit downloads and caches stuff as part of the startup process, it puts it in what we call a local bundle cache folder.

However if you have clusters of users, it doesn't make sense that every single user is downloading the same thing, in this case, you can set up caches that are shared to avoid redundant downloads.



Not every facility has access to the internet, usually because of security restrictions, so there are also ways to distribute configurations which are fully self contained.
In this case, you start with the config
Instead of uploading it to shotgun, you pass it to a build process which will cache all its dependencies and create a self contained bundle
This is then uploaded to Shotgun
When a user is running shotgun desktop, it simply downloads the self contained bundle from shotgun and that contains everything that's needed.



So far we have looked at this from the outside - shotgun desktop, maya and all the DCCs are somehow - as part of their startup they are all managing the distributed setups, making sure that before toolkit is running properly, the latest config has been downloaded, all dependencies have been downloaded etc.

This is all handled by an API - called the bootstrap API or ToolkitManager - and it's very straight forward to use.

You can have for example a render farm python script start up, grab a config from Shotgun and retrieve the config and its dependencies. Tools or automation can then be executed.

Remote Compute / Bootstrap API

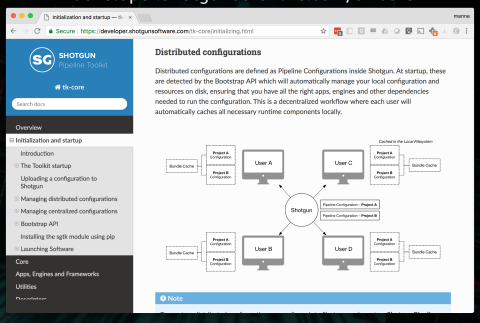
```python
import sgtk

# Start up a Toolkit Manager
mgr = sgtk.bootstrap.ToolkitManager()

# Set the base configuration to the default config
# Note that the version token is not specified, meaning that
# the latest version will be looked up and used
mgr.base_configuration = "sgtk:descriptor:app_store?name=tk-config-basic"

# Each bootstrap session is normally defined by a plugin id. We recommend
# using a 'basic.' prefix if you want your bootstrap session to pick up the
# standard toolkit configurations
mgr.plugin_id = "basic.my_toolkit_plugin"

# now start up the maya engine for a given Shotgun object
e = mgr.bootstrap_engine("tk-maya", entity={"type": "Asset", "id": 1234})
```

AUTODESK

Here's an example from our dev docs

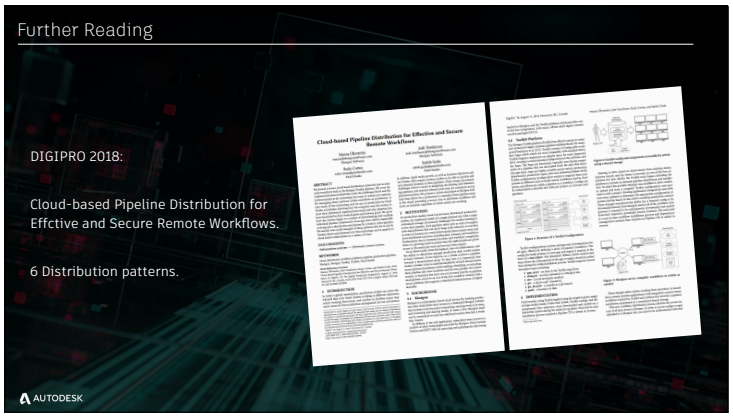
Further Reading

developer.shotgunsoftware.com/tk-core

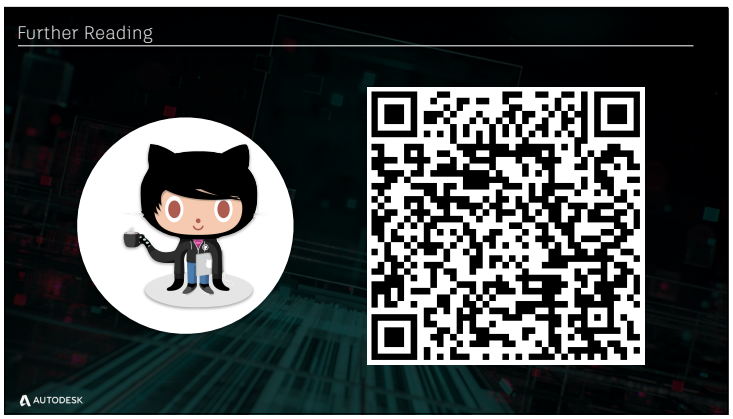


AUTODESK

In general, I also recommend checking out our developer docs – they contain a bunch of detailed information about distributed configs.



On saturday, we also presented a paper with a series of patterns or recipes all based around distributed configs.



You can find links to the API documentation, to our digipro paper, these slides in pdf format etc by following the QR link here!

Do you have any questions?