* configuration management
* orchestration
* manage infrastructure at scale using infrastructure as code principles.

**Quote** guiding principle.

make it work.

Make it right

make it fast.

ansible falls into a broad category of tools that I would consider infrastructure automation. And really that's how do we get this software this infrastructure stood up in a way repeatable reliably using some tooling that we can commit to repository and have to use later.

what's special about configuration management tools like ansible puppet and chef compared to all of these other scripting or code libraries that you might use???

* it's because configuration management provides you with some abstractions that help you do that better. Right and I say do that being make changes to for structure with confidence.
* So you get things like cross-platform implementations were kind of the nitty gritty details are hidden for you. You get templating in variables and other control structures for applying logic to the changes you need to make. It gives you sort of a framework for building modular code that you can reuse that's encapsulated well .
* And in the end though I think one of the biggest things you get out of it is this concept of **idempotents** which is if you need to make a change to an environment you can execute something. And if it's idempotent you can execute it again and nothing changes in that second run. So the first time you run something it takes you from whatever the current state is to the end desired state. Then you can rerun it and trust that no other things no other actions will take place.

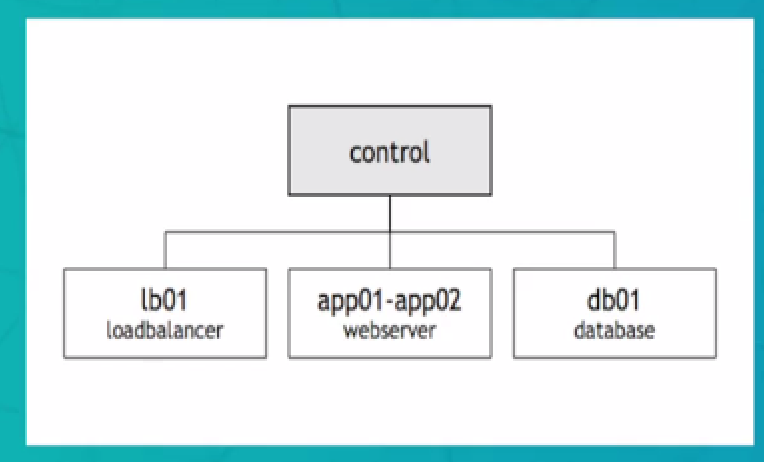
And so you get safety out of this ability to codify your changes and sort of in and declared a state as opposed to always execute these same actions over and over again. And so having idempotents built into these tools is really empowering you to have confidence that you can run them very often. And it's in that repeated running that you can build up confidence and the state of your infrastructure.

**Configuration Management** is really about taking a particular host and getting it into its and desired state. So we don't necessarily know where it is now but we can declare the desired end state and it config management tool is really good about taking you from where you are to where you want to go.

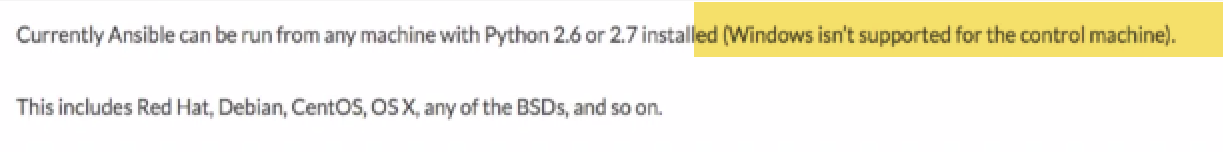
There is still this question though of how do you do that across many hosts. Right. And if you think about complex architecture there may be web hosts and database hosts and look balancers and the order through which you apply the changes to those different tiers really matters.

So configuration management tools really focused on the content of a single server but somebody has to be in charge of making sure that before you update your web servers you mix your database scheme as applied first. Those types of changes and that's really the domain of the orchestration tool.

**Preparation**

we're using is a common three tier web application architecture. We have a load balancer tear along with the application tier and databased here. 

in addition to these three common tiers, we have a control machine that sits off to the side and the control machine really acts as the orchestration of the place that we're going to install and spool and do all of our mission management throughout this course.



you cannot use Windows as the control machine. So ansible supports Windows as a target

**Installation**

**02 Foundation**

**005 Inventory Pt 1**

Now with ansible every command has both the content of the change you want to make or the command that you want to run. But also the target that you want to run against with the ansible tool…. you have to specify all of your potential targets in advance. And the way that you do that is through populating an inventory.

Inventory can take a 2 forms. 1.) simple static file that has a list of hosts.

2.) dynamic inventory which is normally implemented as a script that calls some other API or service to populate all of the host information and variable data.

inventory lists the hosts but it can also list additional information alongside those hosts names. And mostly those extra details are around how you connect to the host. So if you're using a non-standard ssh port or a special hostname to connect to the host you can specify that in the inventory and so will also give you the ability to specify that on the command line per invocation. But by putting it into the inventory you can trust that whenever you connect to that host the right parameters for that host will be used.

Inventory also allows you to group hosts together by role and you can then use those groups as part of your target's one and executing commands

you can also provide a variable values and past variable data in the inventory.

Now adding our own hosts we could put them back into this file and then continue working with ansible. The problem with that is that we would then have to edit this file that's on the control hosts in the system location every time we want to make a change to the inventory and we can't keep those posts in a file located in our local directory and get repository.

**So part of managing our infrastructure is code really is to use a process that allows us to commit every change into a repository and track that changes to that inventory along with the ansible playbooks and the roles and all of the changes we want to make to our infrastructure.**

So rather than editing the system default at the /etc/ansible/hosts … My recommendation is to create a local inventory file and populate that inventory file with your hosts and keep that next to your playbooks and your roles inside of your git repo. And that way you can track the changes over time.

**006 Inventory Pt 2**

In our local directory, create an ansible folder and inside of this ansible folder -🡪create a file Dev and populate the hosts.

Now the format for the inventory file can be just a simple list of names

You can also group posts together into arbitrary groups.

In our case we're going to use the host's role or function and in our scenario as their group name.

So we'll create a group for the load balancer (lb01) , group for the web servers and a group for the databases and also group for the control host.

So in our file Lets start by creating the load balancing group and that's done by simply putting the name of the group inside of square brackets and underneath that the name of the host LBO one. And then we'll leave a new line and create a new group web server and the web server has App01 App02 . We have our database group with TBO 1 and our control group that's important not to forget the control server we want every machine in our environment to be controlled and managed using the exact same process even the machine that's doing the control. So there's an initial bootstrap that's required when are installing and spel that can't be done with ansible before ansible is installed. But once we have ansible installed we want to use the ansible process to manage the rest of any configuration that's required even on our control server. So we manage every host the exact same way. So now that I have all of these hosts filled out I'm going to save this file and just call it. You can name it whatever you'd like for your local environment. So now if we go here and type ansible list hosts all we still see that nothing is matching because we're not passing this inventory file that we created. So the way that ansible allows us to do that is by passing a dash parameter when we execute a command . So if we go into that Ansdell directory We were just in we can type ansible dash Hi dev lists hosts all. And now we see the host we've populated a few things to note about this. One is the control host is just entered by hostname and by default ansible it is going to attempt to S-sh into every host that's can be in the story since we're going to execute ansible from the control host by default. It's going to attempt to S-sh into itself which if you set up all the permissions and the key trust that will work but it's it's unnecessary because we're already on that host. So ansible gives us the ability to specify the connection type. In this case we're going to change the ansible connection to local type which tells ansible just to connect without attempting to create an S-sh section first. And that's just for the control since that's where we'll be executing the commands from. And that should have no impact on our ability to execute. One last optimization is to address the fact that we have to now pass dash Hi-Def with every command . The precedence order that ansible follows is to first looking at c ansible hosts. If it's not provided any other inventory information and that's set by the global Ansible configuration file which is an the handsful ansible that TFG. So here you see in this line inventory is at the ansible hosts and ansible allows us to override this configuration by putting a file on our local directory and we'll use that to set the default tune Torri to our local def file so that we don't have to tap type. That's the idea of every time. So we'll create another file and it will create the default section in that file and pass inventory is the def file that's in our local current directory. We will save that as ansible that CFT in our ansible directory. Now here in our Ansible directory We have our ansible CFT file. And if we run ansible list hosts all without passing Dasch I've dev. It's now picking that up by default using that local override in the configuration file. And we now can execute commands just like we would if we're using the global inventory but we have the added benefit of being able to track or inventory in our code repo locally with our playbooks. So with that we have our host set up next we'll look at different ways that we can select subsets of that inventory if we want to target particular Hosa particular host groups.