



Automating Your Automation The Care and Feeding of Jenkins

Jeff McKenzie • jeff.mckenzie@insight.com • @jeffreymckenzie

Good morning everyone –
Thanks for being here.....

We're going to be looking at Jenkins
-- brief intro
-- how to care for it

My name is Jeff McKenzie,
And I am a Practice Manager for App Dev and Infrastructure
At Insight Digital Innovation in Columbus Ohio
We used to be Cardinal Solutions
But acquired in August 2018 by Insight

Creating meaningful connections that help
businesses run smarter.



Insight is a global, fortune 500 company

-- does a lot of things in tech

But, acquired Cardinal to help expand their
Digital Innovation division

Digital innovation solves business problems using:

- Custom development
- With established as well as emergent technologies

Digital Innovation Capabilities



- Do a lot of cloud work, app modernization
- Big data, predictive analytics
- Devops on both the Microsoft and open source side
- As well as a fair amount of IoT solutions

Award winning technology

Our combined IT industry knowledge and technology expertise have earned us numerous Microsoft honors through the years.



2019

2019 Microsoft U.S. Intelligent Cloud Partner of the Year – App Innovation
2019 Microsoft U.S. Partner Choice Award Winner – Data and AI

2018

2018 Microsoft Worldwide Partner of the Year for Open Source on Azure
2018 Microsoft Worldwide Partner of the Year for Artificial Intelligence
2018 Microsoft Worldwide Financial Services Partner of the Year Finalist
2018 Microsoft U.S. Partner of the Year for Dev Ops
2018 Microsoft U.S. Partner of the Year for Internet of Things

2017

2017 Microsoft Worldwide Partner of the Year Winner for Mobile App Dev

2016

2016 Microsoft Worldwide Partner of the Year for Internet of Things



Azure
Expert
MSP



As a worldwide Microsoft partner, Insight has received a good share of awards,

Including: IoT Partner of the year in 2016

Mobile App Dev and Open Source Azure in 2017

AI and Modern Workspace awards in 2018

Azure Expert MSP –

Completed a rigorous application process with Microsoft to verify successful completion

of projects across almost all Azure service offerings

We passed a 300-hour on-site audit by an independent third party

Also have more than 1,000 Azure-focused engineers and service professionals

<https://azure.microsoft.com/en-us/partners/>

@jeffreymckenzie



Today, talk about/work with Jenkins

How many have worked with Jenkins?

What is Jenkins here supposed to be? [butler]

=====

Logo from <https://jenkins.io/artwork>

@jeffreymckenzie



Why a butler?

What does that have to do with Jenkins? [discussion]

=====

Image from <http://www.pensandpatron.com/lists/never-knew-downton-abbey-ob/>

<http://d3f88t1ya8f0ec.cloudfront.net/wp-content/uploads/2018/07/23152933/charles-carlson.jpg>

@jeffreymckenzie



Know your limits, Master Wayne.

What I like about the Butler theme –

- Jenkins does what you tell it
- Will do almost whatever you say
- Up to you to guide/make good decisions about what butler does
- And if you don't...

=====

Image from <https://getyarn.io/yarn-clip/da131918-6565-4be8-91f3-3f9564e9808>

<https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiCycmjNndAhUK04MKHZAiAAcQjRx6BAGBEAU&url=https%3A%2F%2Fgetyarn.io%2Fyarn-clip%2Fd131918-6565-4be8-91f3-3f9564e9808&psig=AOvVaw3mZTtbp4cJsUQWwudTn2T3&ust=1538076646833110>

@jeffreymckenzie



-- end up with unhappy butler.

For those who have used Jenkins, how have you used it?

Basic freestyle point-&-click-jobs?
pipeline?
global libraries?
job-dsl?

=====

Logo from <https://jenkins.io/artwork>

@jeffreymckenzie



i am by no means an expert

-- but I have been using Jenkins extensively for the last couple of years
-- want to share not so much how to use jenkins itself (a little, but not the focus)

but what options you have for improving your maintenance of Jenkins automation,

....basically its care and feeding.

=====

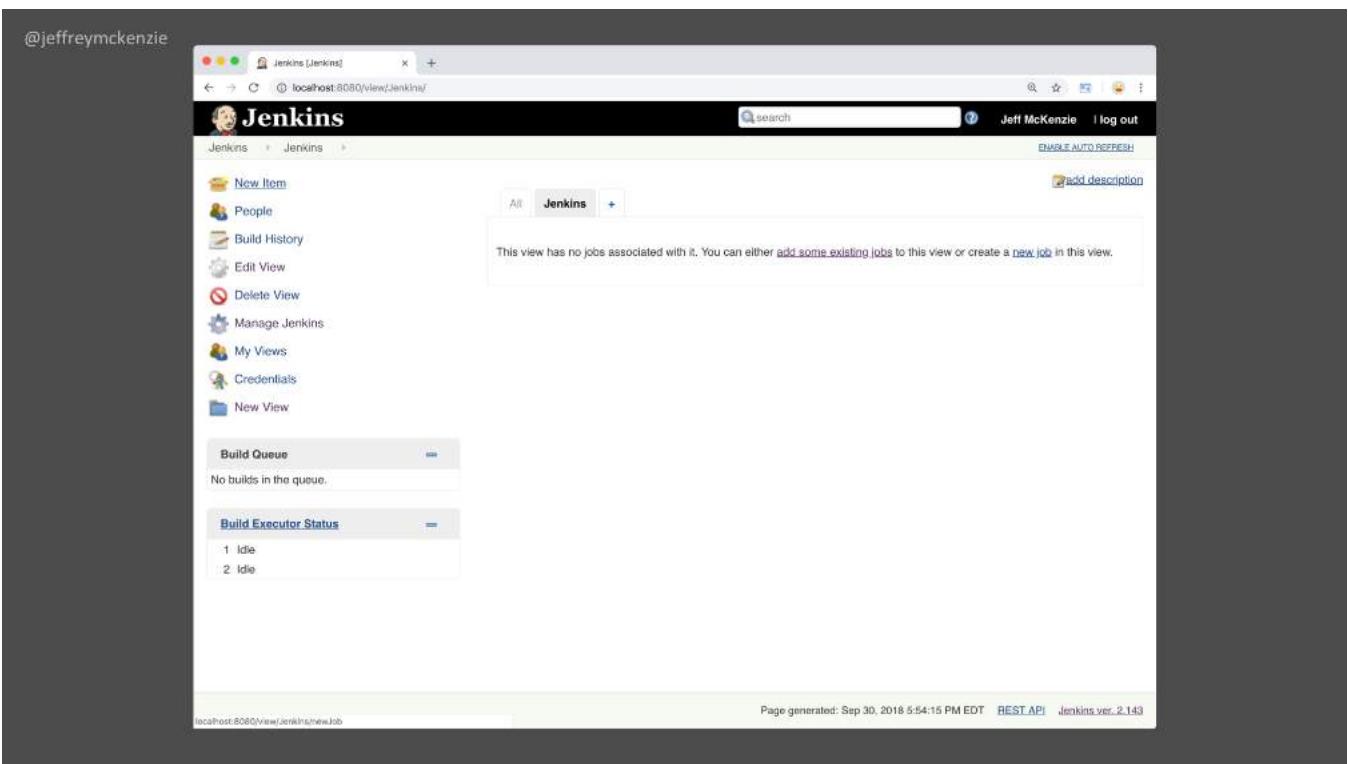
Logo from <https://jenkins.io/artwork>

Job / Project

Basic unit of work in Jenkins

- Difference between job/project
- -- PROJECT, gonna try
- What is a job? examples, *you*
- “will Show example of freestyle job”

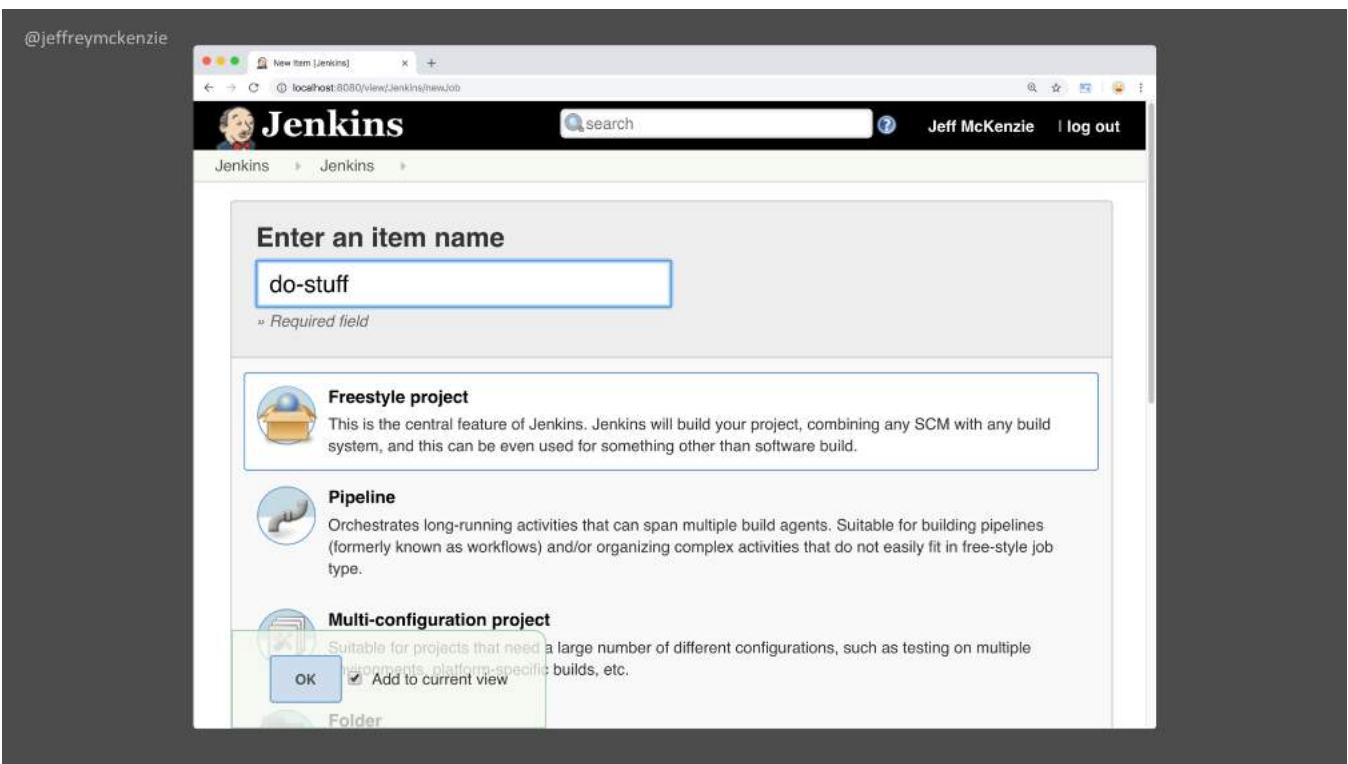
@jeffreymckenzie



-- disclaimer – not a full Jenkins tutorial, but want to set groundwork
In case ppl seeing for first time

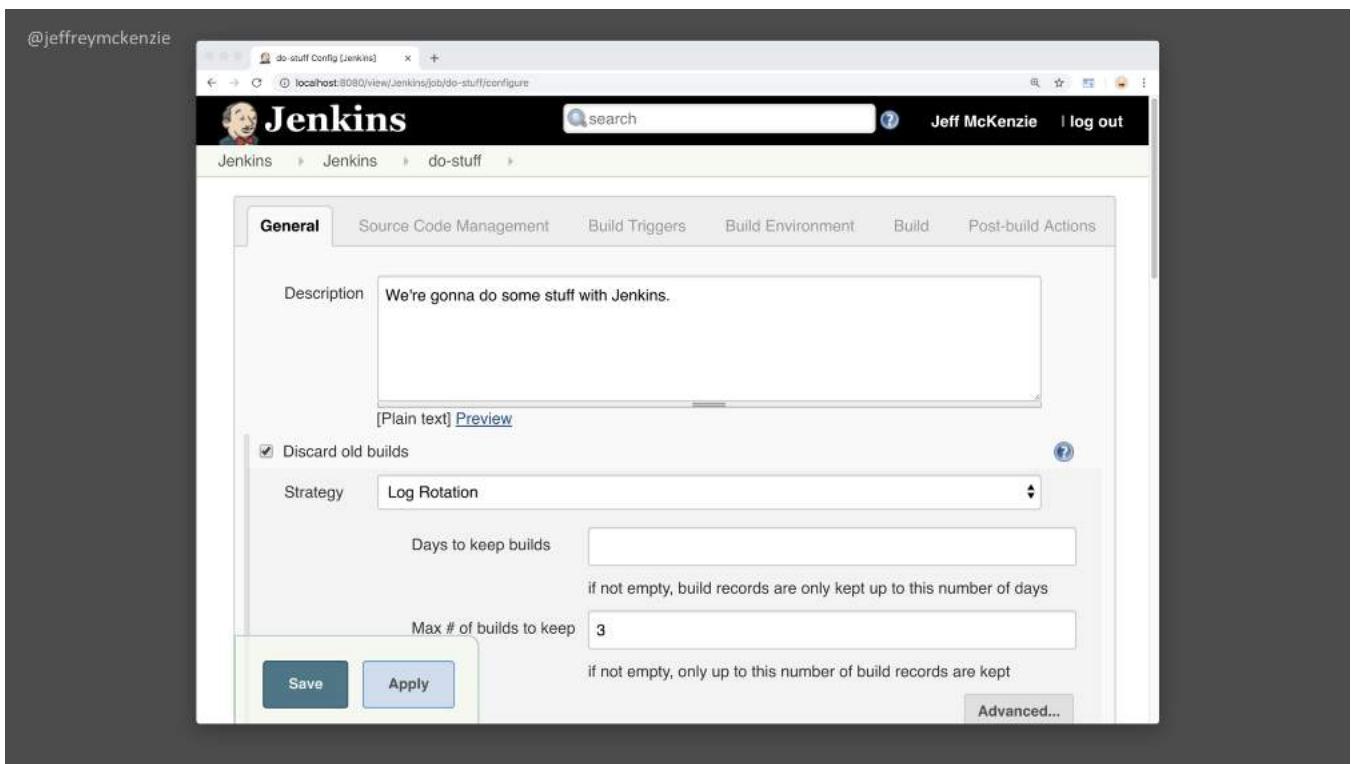
-- after install, default view
-- what you see
-- new item, it all starts with that....

@jeffreymckenzie



-- new freestyle project, most basic

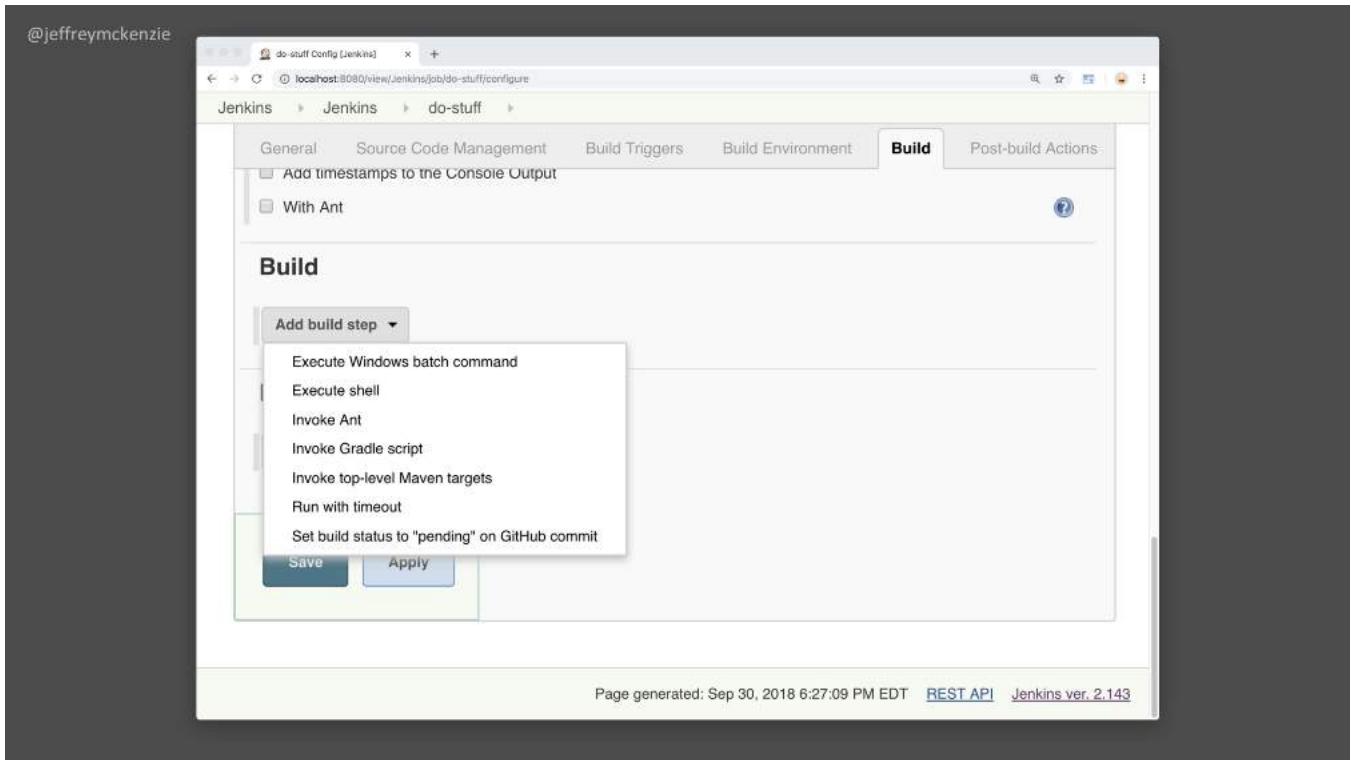
-- name, whatever u want



-- first view/screen, what it looks like

-- description

-- build/log history



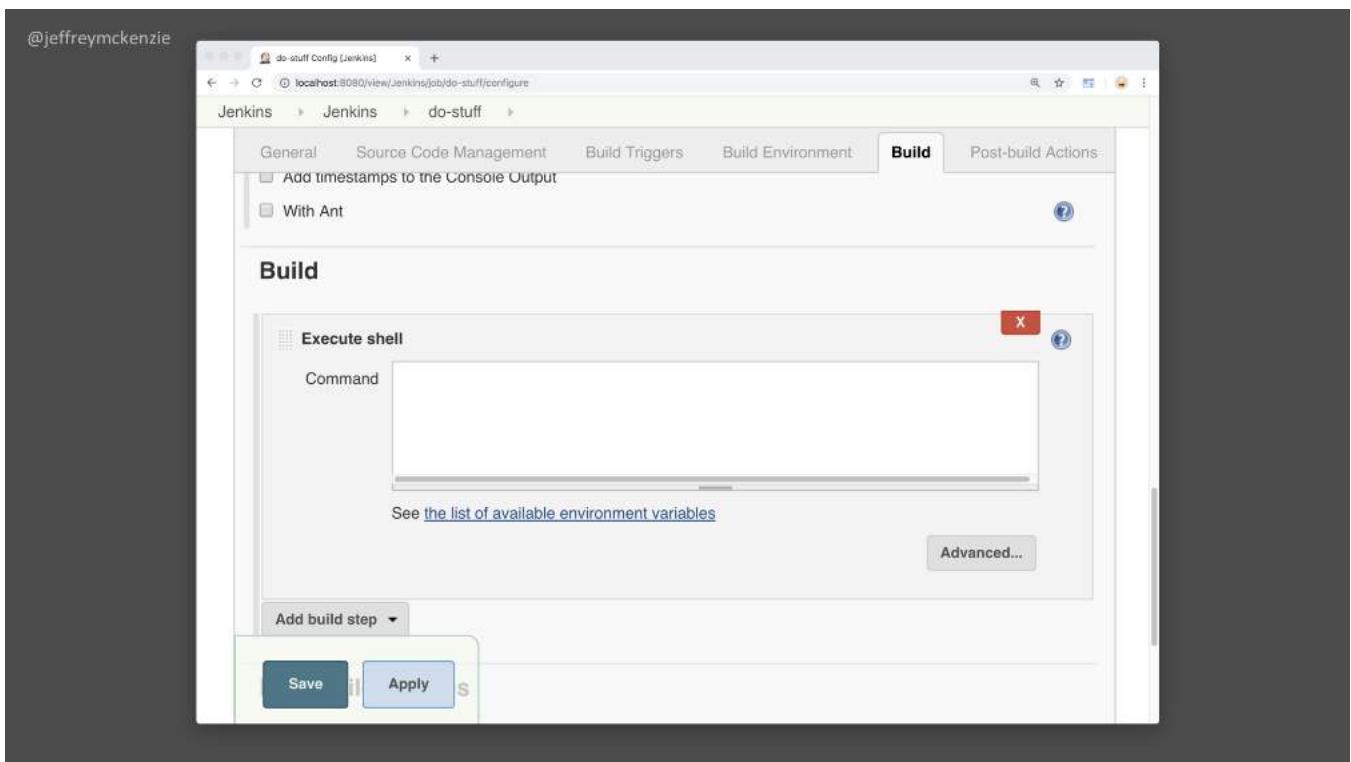
-- jump down to build tab

--add build step

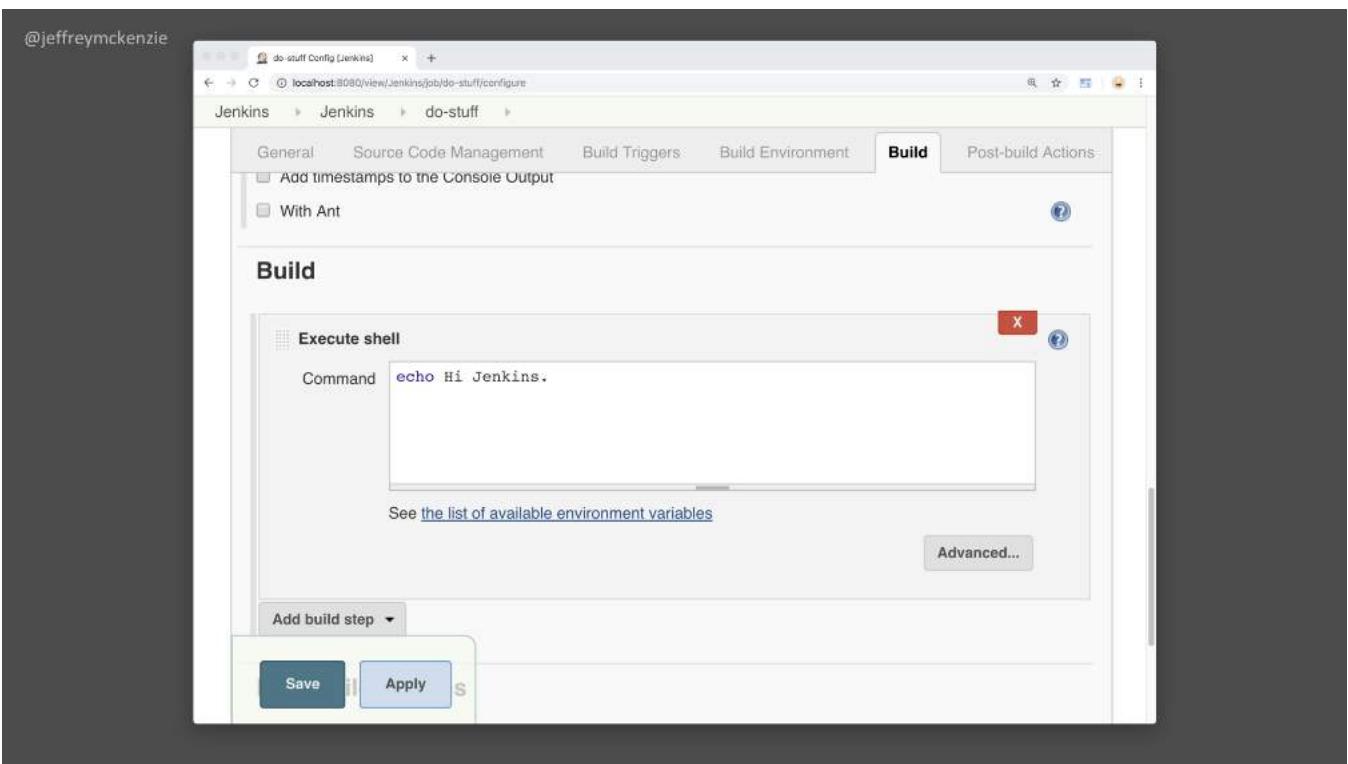
--what want Jenkins to actually do

-- select execute shell

@jeffreymckenzie



There's our single build step



Add a very simple command

Echo, which just means print to the screen

Say Hi Jenkins

@jeffreymckenzie

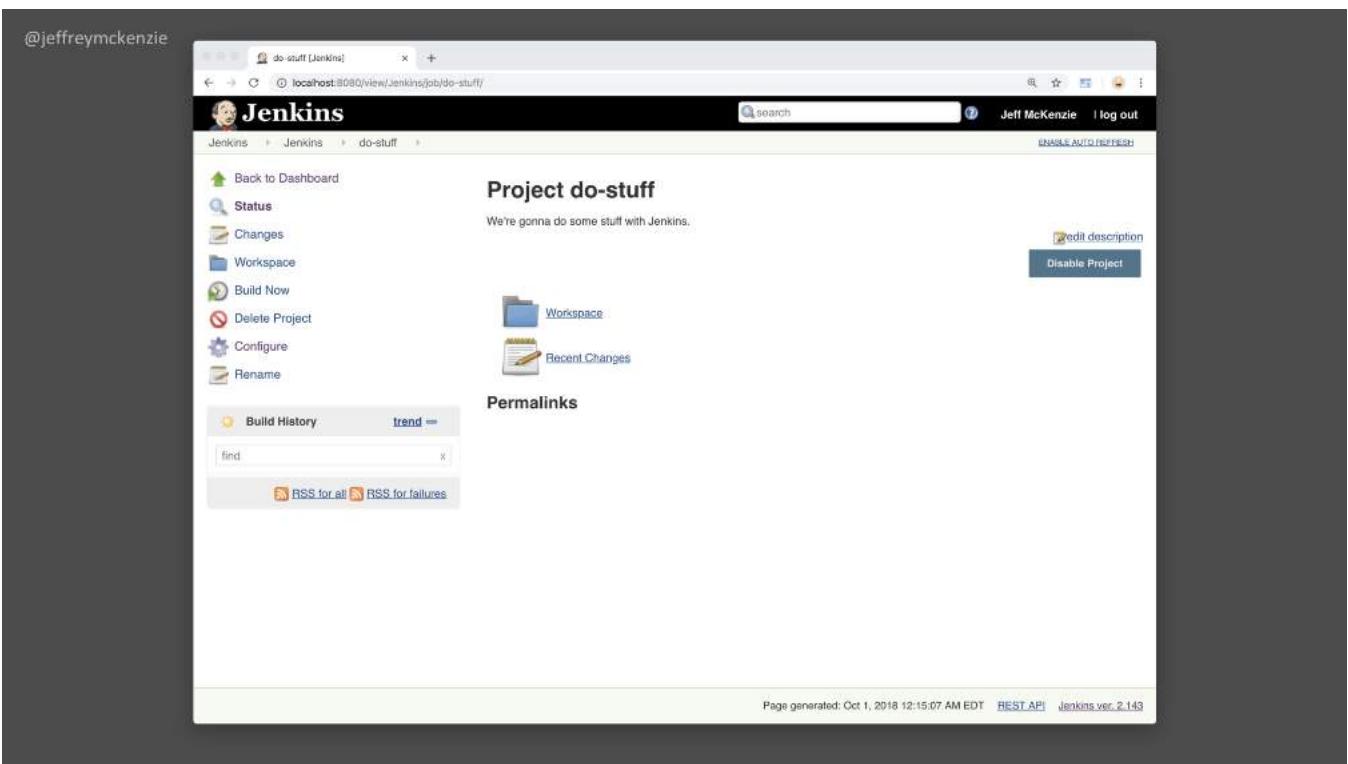
The screenshot shows the Jenkins web interface at localhost:8080/view/Jenkins/. The left sidebar contains links for New Item, People, Build History, Edit View, Delete View, Manage Jenkins, My Views, Credentials, and New View. The main area features a table titled "Jenkins" with columns S, W, Name (sorted by name), Last Success, Last Failure, and Last Duration. A single item named "do-stuff" is listed with a yellow sun icon, indicating it is currently building. Below the table are sections for "Build Queue" (empty) and "Build Executor Status" (2 Idle). At the bottom, a footer bar displays "Page generated: Sep 30, 2016 6:47:00 PM EDT" and links for REST API and Jenkins ver. 2.143.

-- go back to main view

-- hasn't run yet

-- take a closer look

@jeffreymckenzie

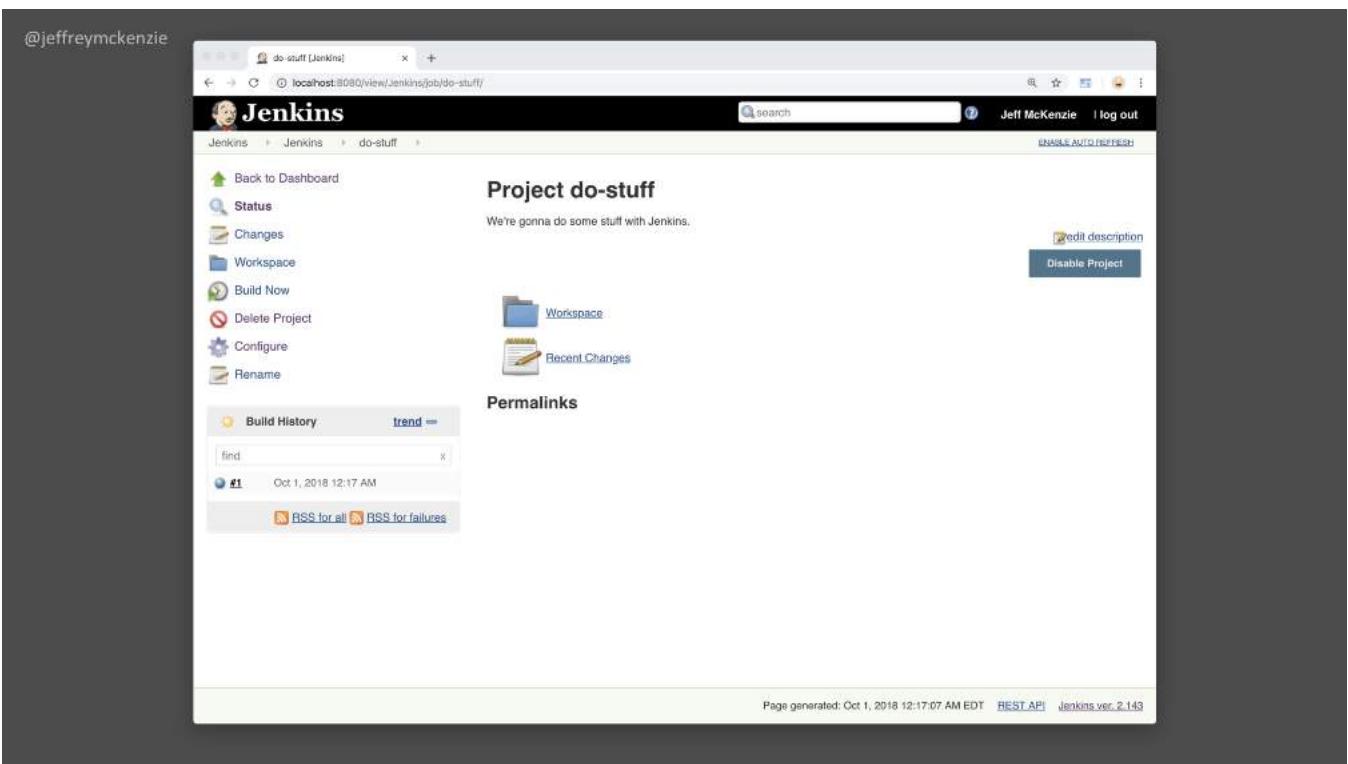


Detailed look at our project

-- on the left, click Build now,

-- which kicks off our job

@jeffreymckenzie



-- and there we have it

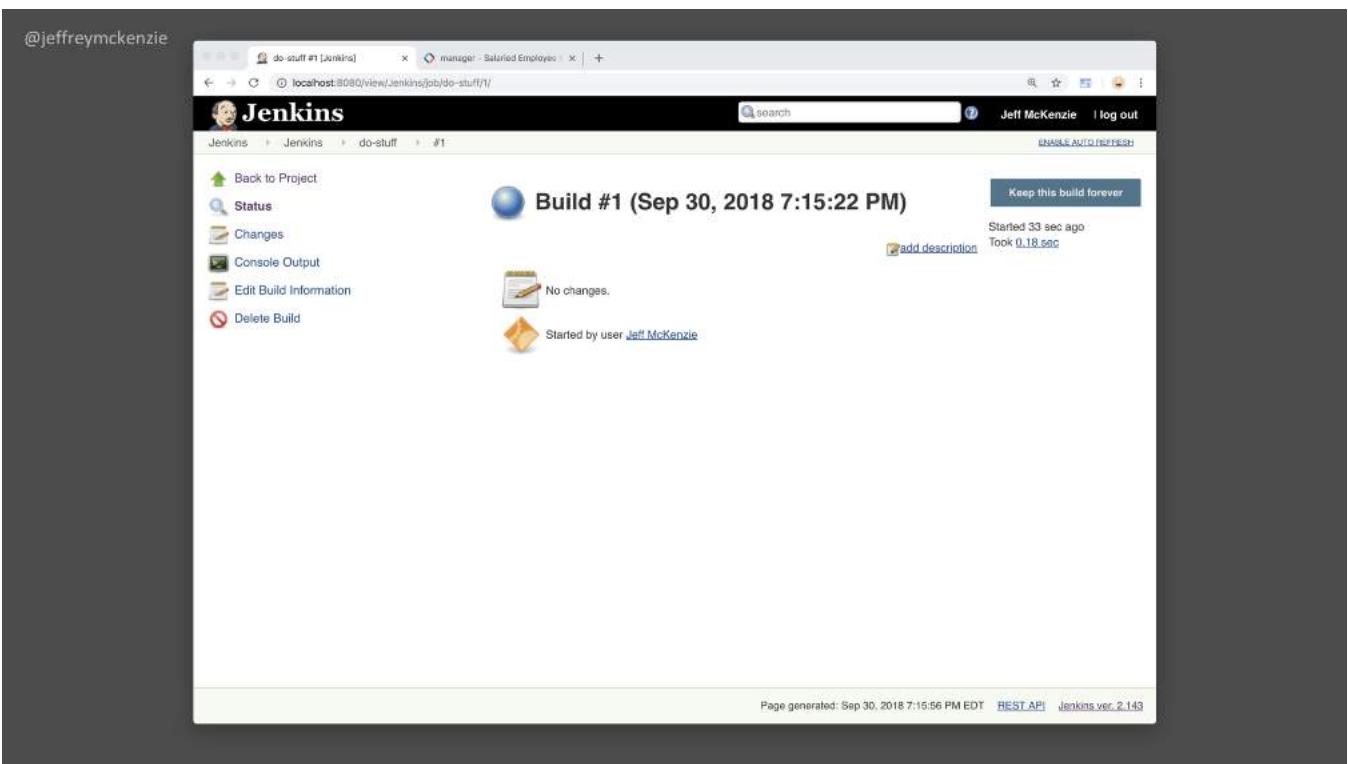
-- you see that, bottom left

[back and forth]

-- might have missed it, tell us job is complete

To verify, can click on the build...

@jeffreymckenzie

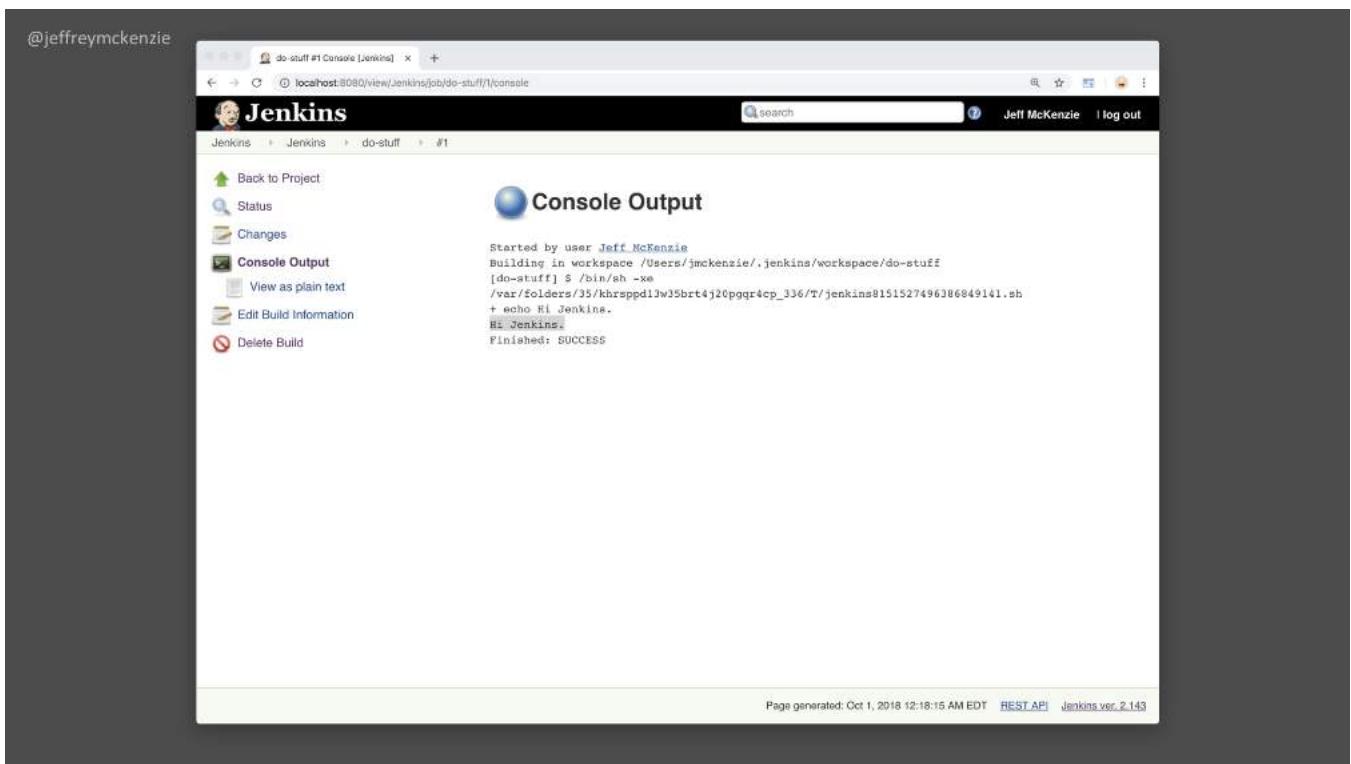


Takes us to information about the result

...then on the left side

Console Input

@jeffreymckenzie

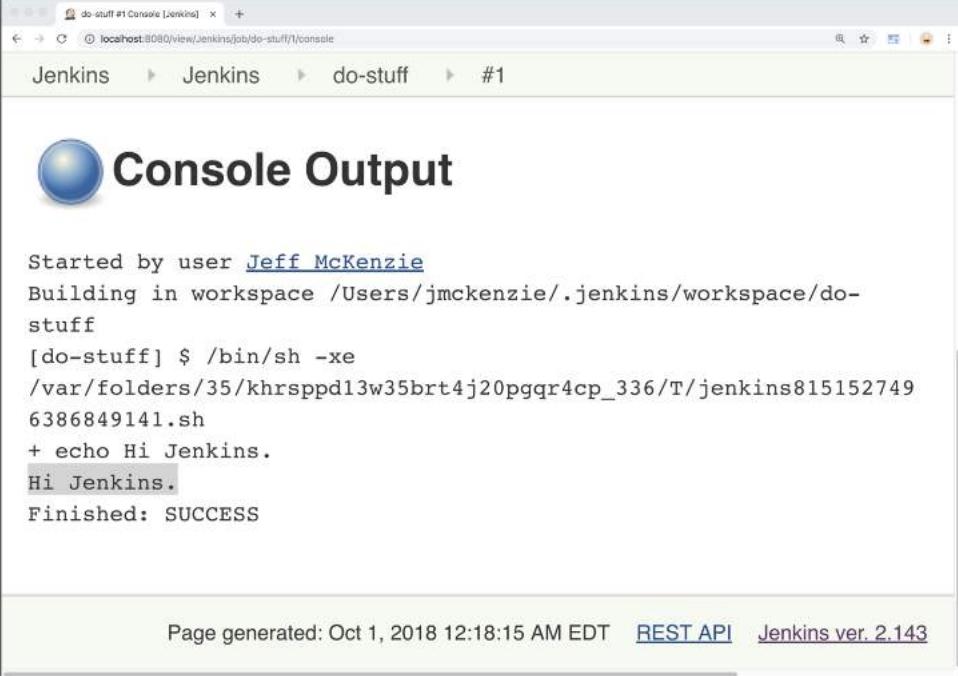


... and there's are result,

Showing Hi Jenkins in the output

...for a closer look....

@jeffreymckenzie



The screenshot shows a Jenkins console output page. The URL is `localhost:8080/view/Jenkins/job/do-stuff/1/console`. The page title is "Console Output". The log output starts with "Started by user Jeff McKenzie" and "Building in workspace /Users/jmckenzie/.jenkins/workspace/do-stuff". It then shows a shell command being run: "[do-stuff] \$ /bin/sh -xe /var/folders/35/khrsppd13w35brt4j20pgqr4cp_336/T/jenkins8151527496386849141.sh + echo Hi Jenkins." followed by the output "Hi Jenkins.". The log concludes with "Finished: SUCCESS". At the bottom of the page, it says "Page generated: Oct 1, 2018 12:18:15 AM EDT REST API Jenkins ver. 2.143".

```
Started by user Jeff McKenzie
Building in workspace /Users/jmckenzie/.jenkins/workspace/do-
stuff
[do-stuff] $ /bin/sh -xe
/var/folders/35/khrsppd13w35brt4j20pgqr4cp_336/T/jenkins815152749
6386849141.sh
+ echo Hi Jenkins.
Hi Jenkins.
Finished: SUCCESS

Page generated: Oct 1, 2018 12:18:15 AM EDT REST API Jenkins ver. 2.143
```

...zoom in a little bit

Tells us that I started it

-- workspace: location on the server

-- shows the command: using shell to execute

...a temporary script it created from my command

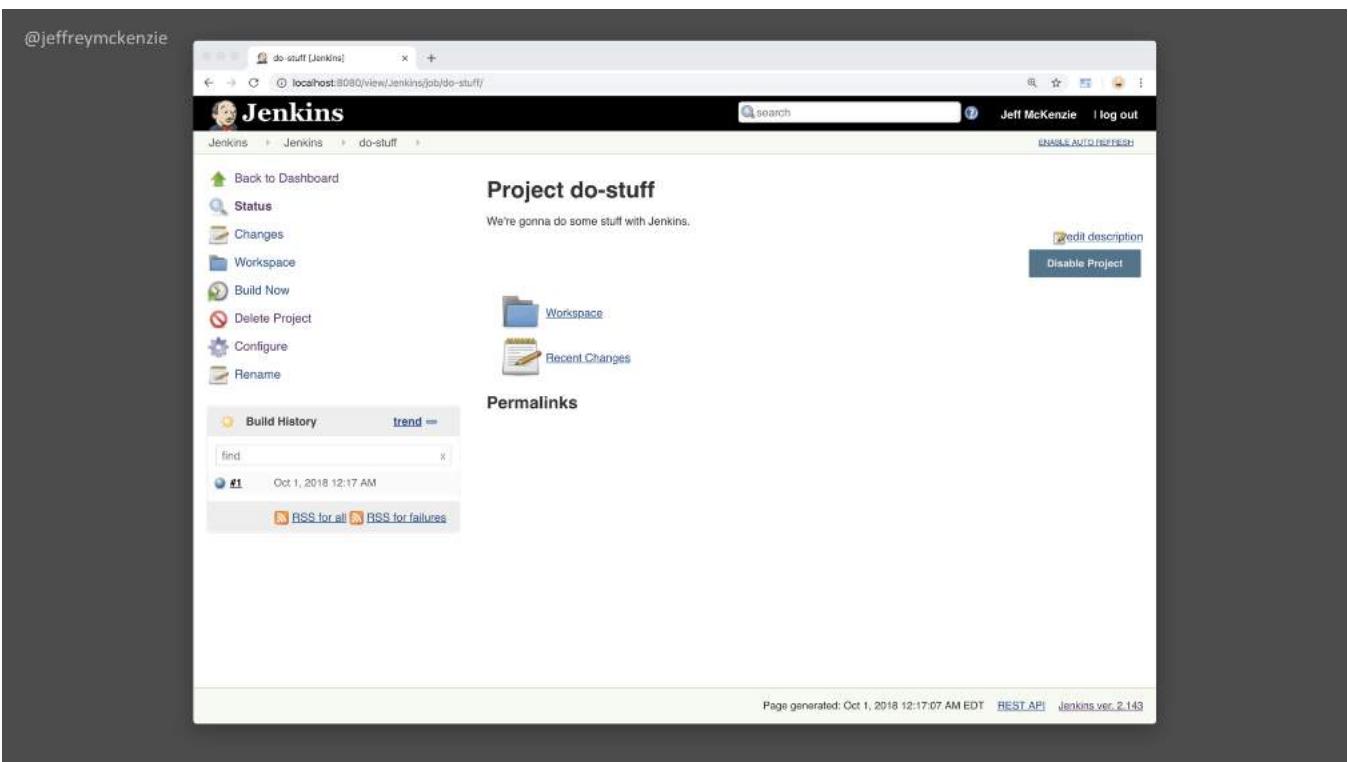
-- shows the command

-- then in the highlighted portion shows the output

-- and shows us the result, which is SUCCESS

Did what supposed to do

@jeffreymckenzie



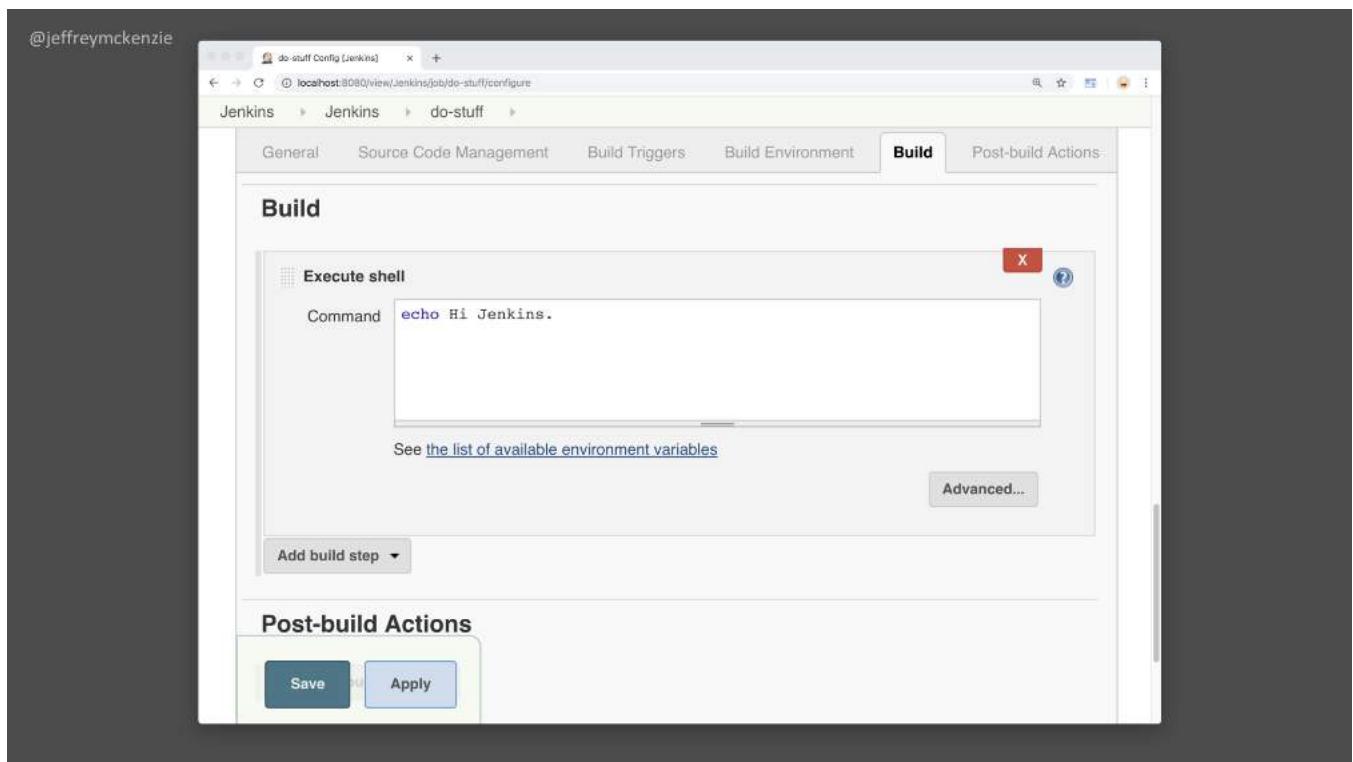
Now what if it had failed?

Let's take a look at that.

Back on our project/job page

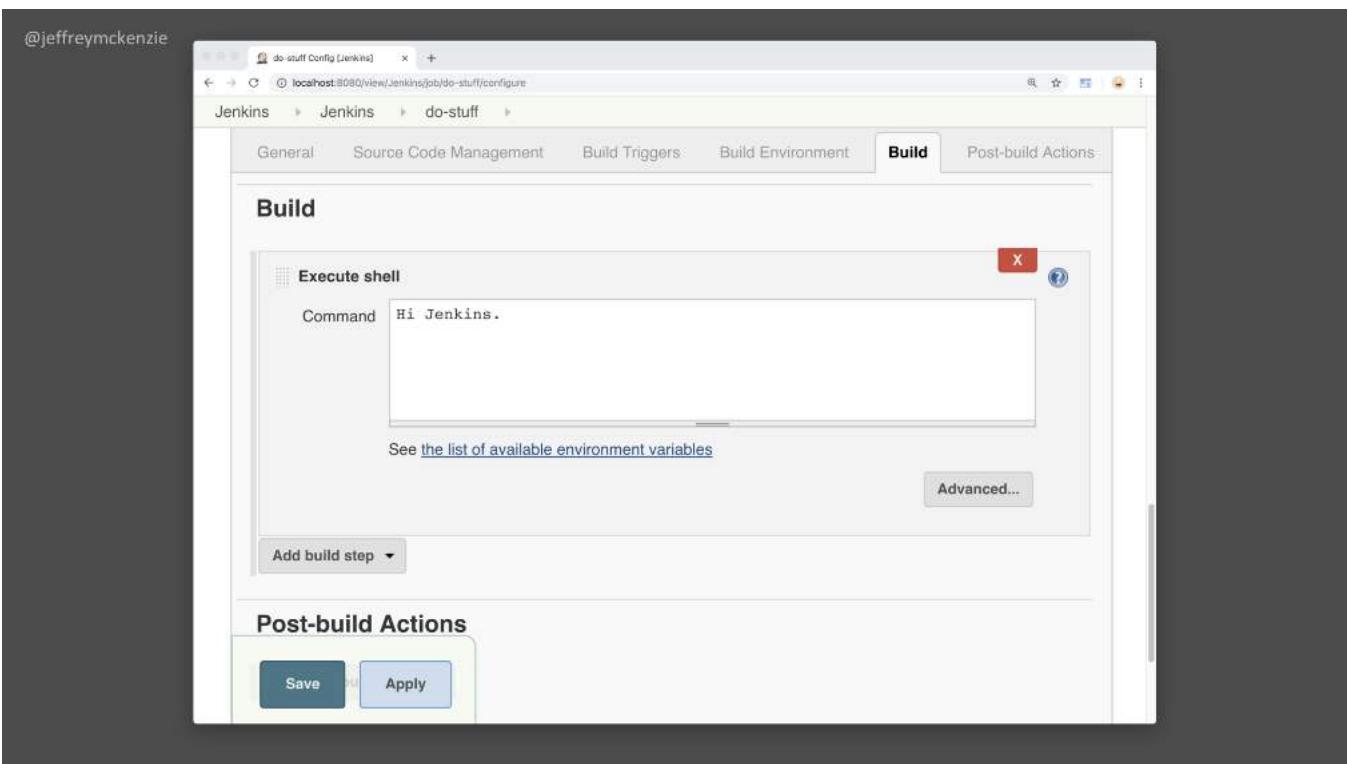
Click configure and go back to our build step

@jeffreymckenzie



Let's remove the echo command from here

And see what happens



Ok we've taken it out...

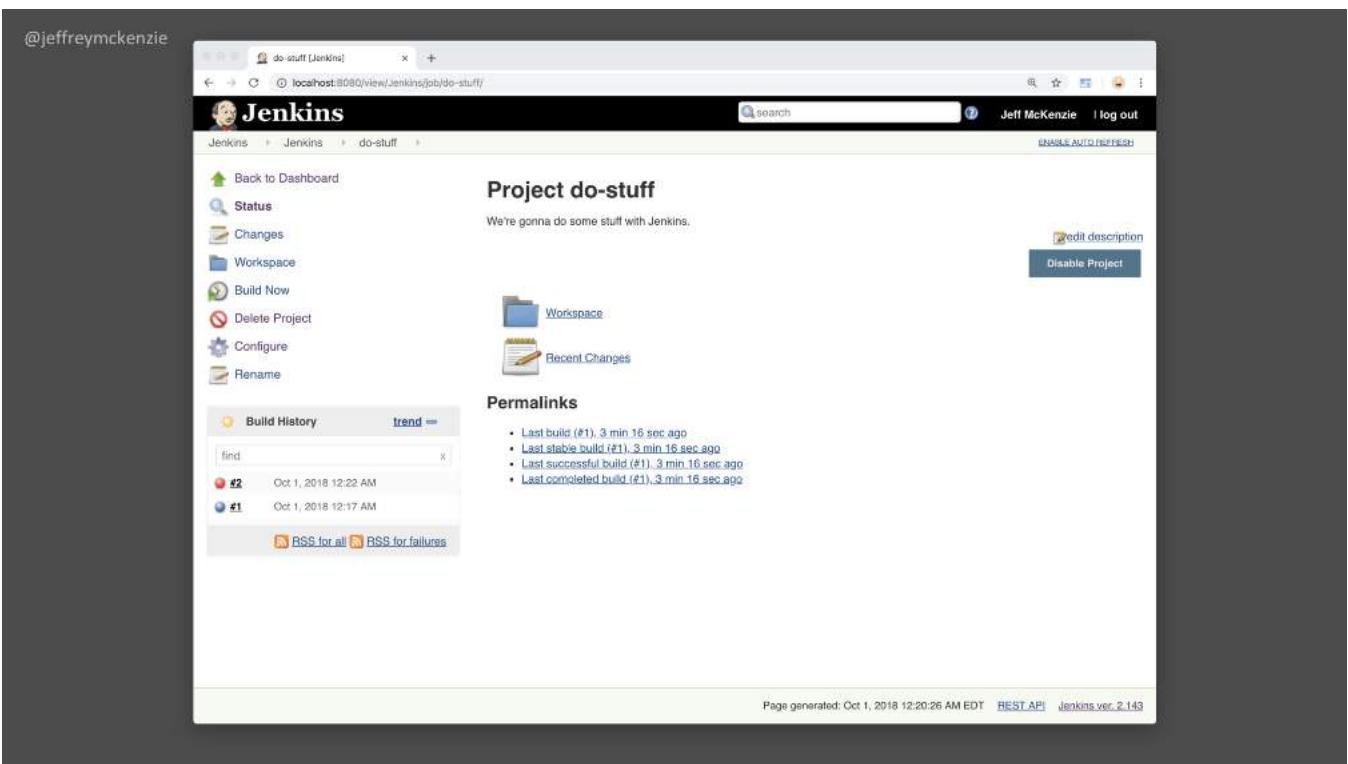
Let's click save and go back to the main page

@jeffreymckenzie

The screenshot shows a Jenkins project page for 'do-stuff'. The left sidebar contains links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, and Rename. The main content area is titled 'Project do-stuff' with the subtitle 'We're gonna do some stuff with Jenkins.' It features two buttons: 'edit description' and 'Disable Project'. Below these are sections for 'Workspace' (with a link) and 'Recent Changes' (with a link). A 'Permalinks' section includes a 'Build History' table with one entry: '#1 Oct 1, 2018 12:17 AM'. At the bottom, there are RSS links for 'RSS for all' and 'RSS for failures'. The footer indicates the page was generated on Oct 1, 2018, at 12:17:07 AM EDT, and shows REST API and Jenkins version 2.143.

Click “build now” again...

@jeffreymckenzie



And we've got a failure

Let's go right to the console output

@jeffreymckenzie

The screenshot shows a Jenkins console output page. The title bar says "do-stuff #2 Console [Jenkins]". The URL is "localhost:8080/view/Jenkins/job/do-stuff/2/console". The page title is "Console Output". A red circular icon with a white exclamation mark is displayed. The log output shows:

```
Started by user Jeff McKenzie
Building in workspace /Users/jmckenzie/.jenkins/workspace/do-stuff
[do-stuff] $ /bin/sh -xe
/var/folders/35/khrsppd13w35brt4j20pgqr4cp_336/T/jenkins366982912
8958591900.sh
+ Hi Jenkins.
/var/folders/35/khrsppd13w35brt4j20pgqr4cp_336/T/jenkins366982912
8958591900.sh: line 2: Hi: command not found
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

At the bottom, it says "Page generated: Oct 1, 2018 12:22:48 AM EDT [REST API](#) [Jenkins ver. 2.143](#)".

And there's our error –

There's no command called "Hi"

So our greeting was being parsed as a command,

Which doesn't exist.

So there we have a FAILURE

@jeffreymckenzie



So there we have Jenkins at its most basic level.

Jenkins jobs/projects can be run manually,

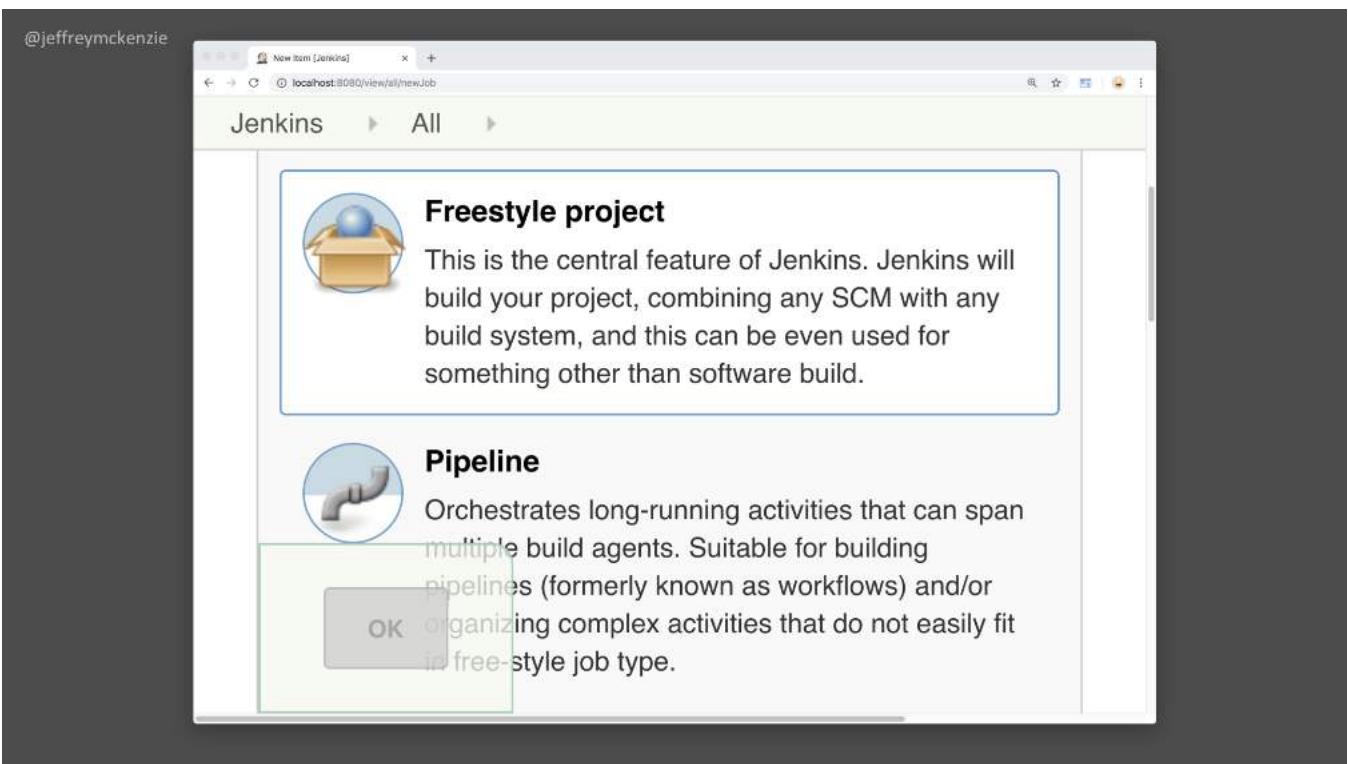
Or can be triggered by an external event.

If you're interested in automation,

You want to interact with Jenkins as little as possible.

=====

Logo from <https://jenkins.io/artwork>

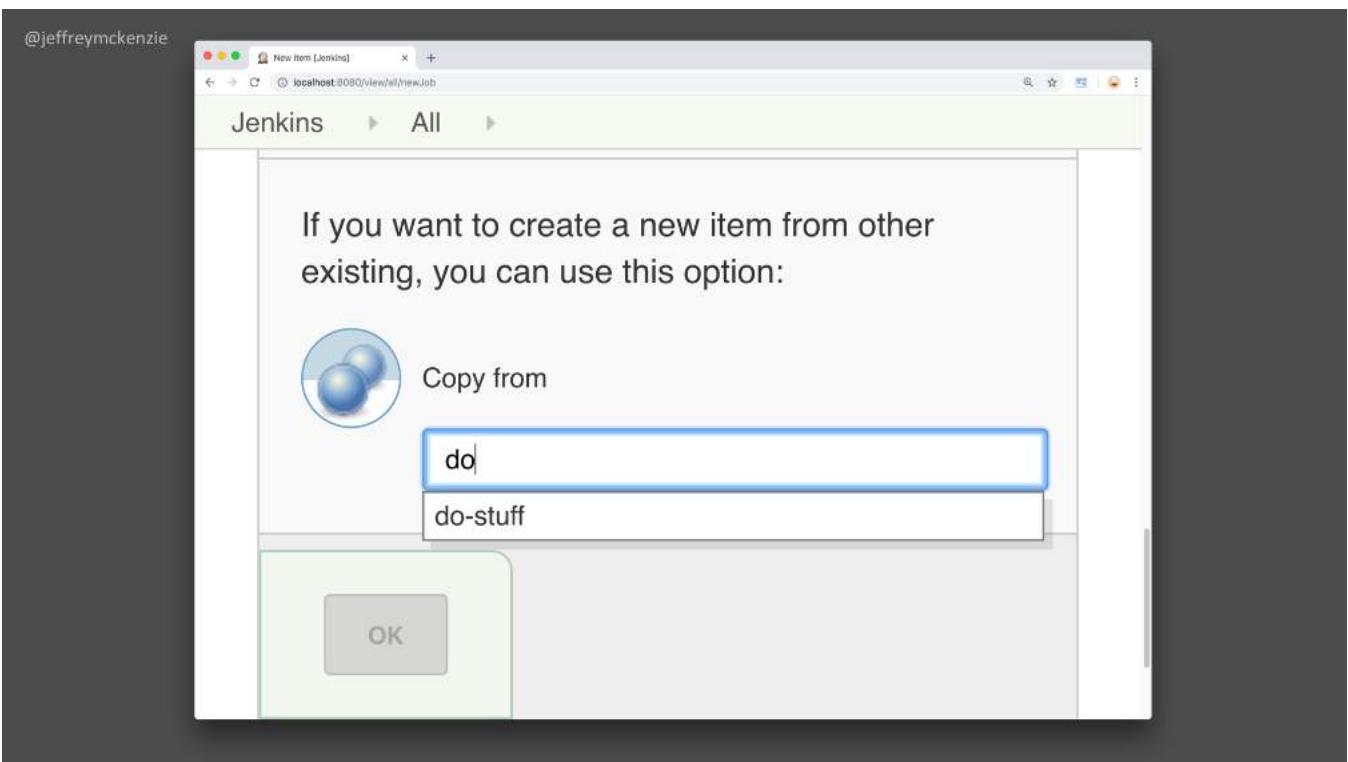


-- Jenkins works great in the beginning, right...

When you are just getting started

But when you add more...

-- [copy] the classic way to scale....



Copy, right?

Who has used this scaling method?

It's great, copies everything from an existing project into new

-- just like copy & paste code...

@jeffreymckenzie



You get spaghetti automation that ties your hands together.

=====

[https://commons.wikimedia.org/wiki/File:Homemade_fresh_pasta_\(spaghetti\)_by_Camille_\(Unsplash\).jpg](https://commons.wikimedia.org/wiki/File:Homemade_fresh_pasta_(spaghetti)_by_Camille_(Unsplash).jpg)

By davide ragusa davideragusa (<https://unsplash.com/photos/FwiLgvi-2Do>) [CC0], via Wikimedia Commons

@jeffreymckenzie



So let's look at a real-life example we can use

To demonstrate ways we can scale Jenkins in a better way.

Imagine an enterprise company

=====

Logo from <https://jenkins.io/artwork>

@jeffreymckenzie



Not just a Fortune 500 company,

But arguably Fortune # 1.

Not in New York, London, or Beijing....

But somewhere a little farther north --

@jeffreymckenzie



Nobody knows quite exactly where it is,

Though people have tried looking...

=====

[https://commons.wikimedia.org/wiki/File:US_Navy_010605-N-0000X-002_Sailors_stationed_aboard_the_Los_Angeles-class_fast_attack_submarine_USS_Scranton_\(SSN_756\)_explore_the_Arctic_ice_surface.jpg](https://commons.wikimedia.org/wiki/File:US_Navy_010605-N-0000X-002_Sailors_stationed_aboard_the_Los_Angeles-class_fast_attack_submarine_USS_Scranton_(SSN_756)_explore_the_Arctic_ice_surface.jpg)

By U.S. Navy photo [Public domain], via Wikimedia Commons

@jeffreymckenzie



A very mysterious operation....

[anybody know what company I'm talking about?]

=====

[https://commons.wikimedia.org/wiki/File:Ever_wonder_what%27s_actually_at_the_North_Pole%3F_Wonder_no_more._\(19676111656\).jpg](https://commons.wikimedia.org/wiki/File:Ever_wonder_what%27s_actually_at_the_North_Pole%3F_Wonder_no_more._(19676111656).jpg)

https://upload.wikimedia.org/wikipedia/commons/d/d3/Ever_wonder_what%27s_actually_at_the_North_Pole%3F_Wonder_no_more._%2819676111656%29.jpg

@jeffreymckenzie



Yes, Santa Claus incorporated,

The man himself...

You want to talk about someone who needs to scale...

This is the case study of all case studies.

=====

[https://commons.wikimedia.org/wiki/File:Flickr_-_The_U.S._Army_-_www.Army.mil_\(144\).jpg](https://commons.wikimedia.org/wiki/File:Flickr_-_The_U.S._Army_-_www.Army.mil_(144).jpg)

By The U.S. Army (www.Army.mil) [Public domain], via Wikimedia Commons

@jeffreymckenzie

Did you know...



Did you know...

Santa employs over 7.2 million reindeer that help him
Distribute all of those gifts on Christmas eve,

=====

[https://commons.wikimedia.org/wiki/File:Reindeer_on_Ikpek_Beach_\(726653378\).jpg](https://commons.wikimedia.org/wiki/File:Reindeer_on_Ikpek_Beach_(726653378).jpg)

By Bering Land Bridge National Preserve (Reindeer on Ikpek Beach)
[CC BY 2.0 (<https://creativecommons.org/licenses/by/2.0>)], via
Wikimedia Commons

@jeffreymckenzie



Santa himself is getting up there,

He's about 1700 years old now.

He used to know all of the reindeers by heart,

Which is important....

=====

[https://commons.wikimedia.org/wiki/File:Saint_Nicholas_Catholic_Church_\(Zanesville,_Ohio\)_-_stained_glass,_St._Nicholas_-_detail,_closeup.jpg](https://commons.wikimedia.org/wiki/File:Saint_Nicholas_Catholic_Church_(Zanesville,_Ohio)_-_stained_glass,_St._Nicholas_-_detail,_closeup.jpg)

By Nheyob [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0>)], from Wikimedia Commons

@jeffreymckenzie



For keeping in contact with all

Of the reindeer and elves,

Keeping everything coordinated

=====

https://commons.wikimedia.org/wiki/File:Kadena_celebrates_holidays_with_Tinsel_Town_151212-F-ZC102-550.jpg

By Senior Airman Omari Bernard [Public domain], via Wikimedia Commons

@jeffreymckenzie



For the night of the Big Dance

On December 24.

So what does Santa do?

=====

https://commons.wikimedia.org/wiki/File:Defense.gov_photo_essay_071203-D-0653H-234.jpg

By English: Linda Hosek [Public domain], via Wikimedia Commons

@jeffreymckenzie



He uses Jenkins, of course.

So let's look at an example of what Santa's going to do here.

-- create new freestyle job called Rudolph...

=====

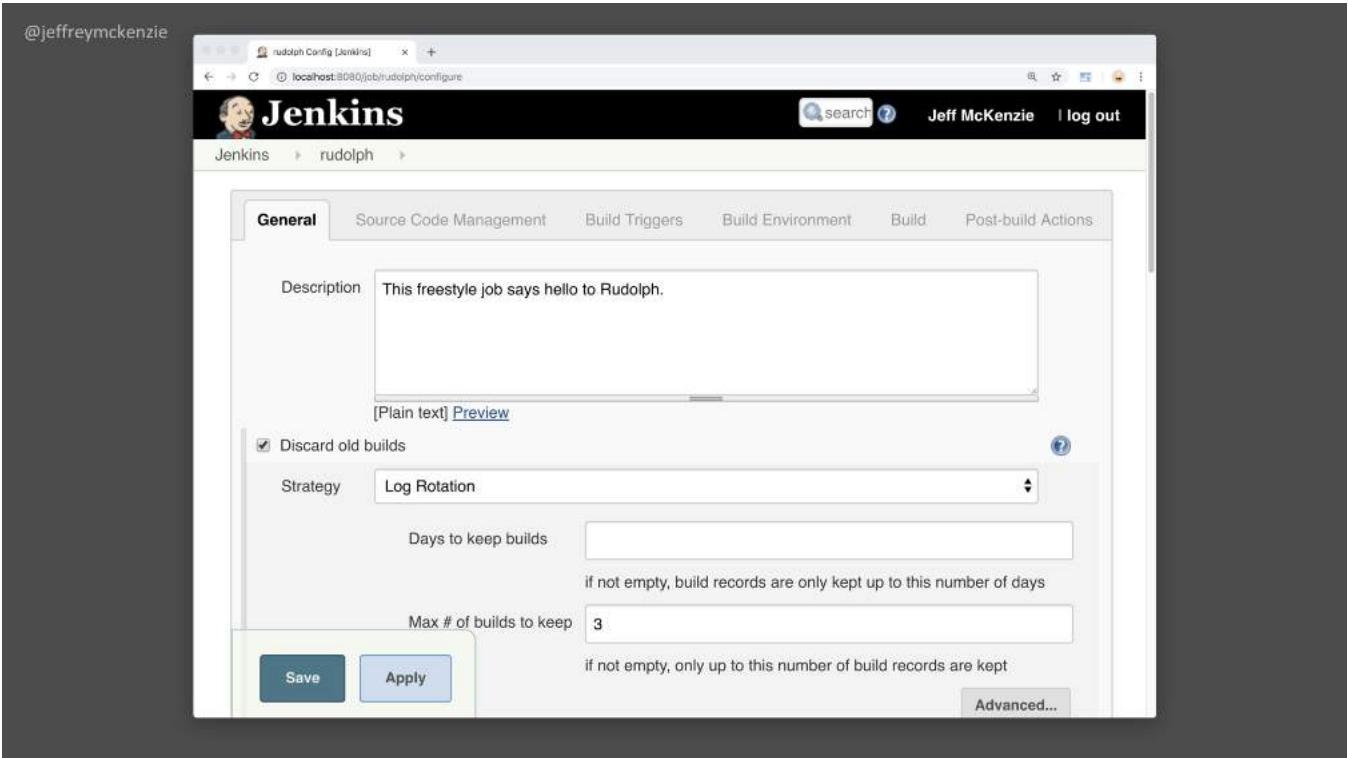
Logo from <https://jenkins.io/artwork>

Created by https://twitter.com/ks_nenasheva

@jeffreymckenzie

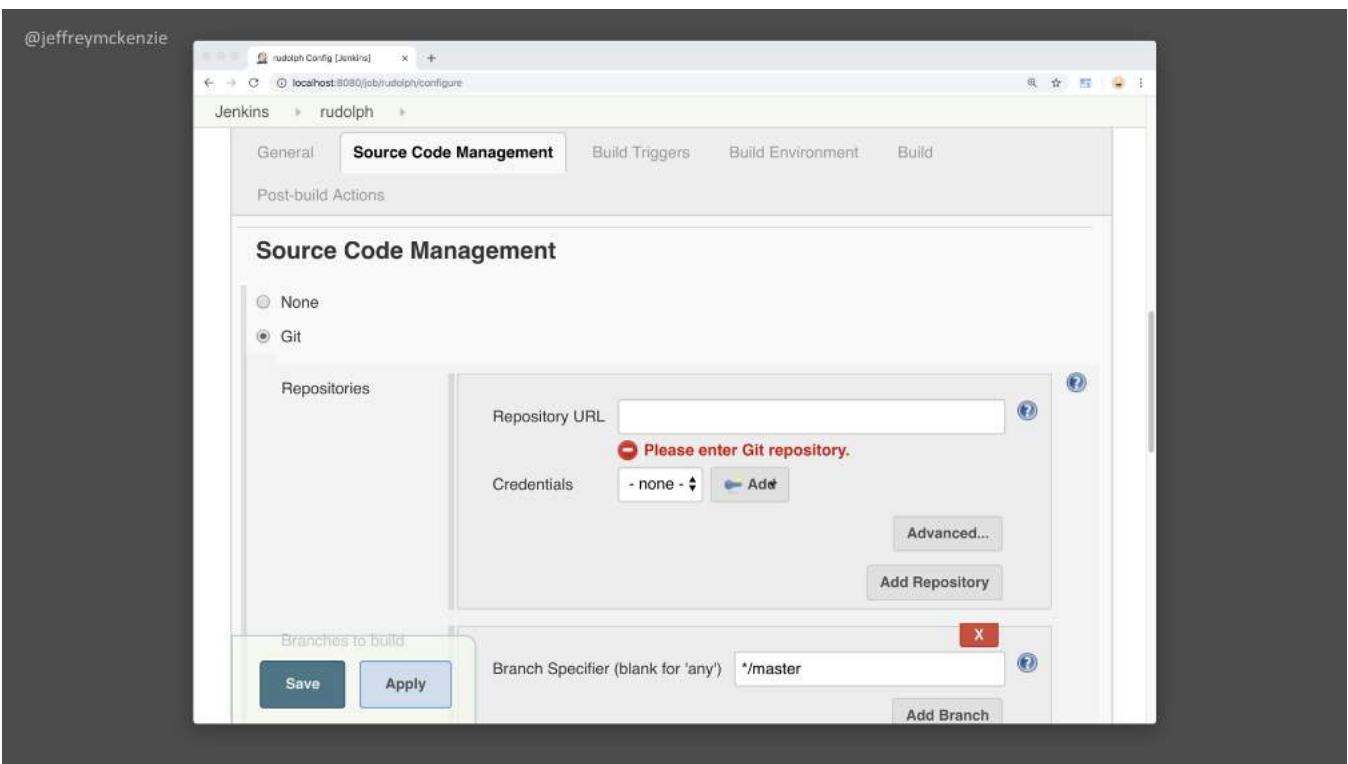
The screenshot shows the Jenkins web interface for creating a new item. The title bar says "New Item [Jenkins]". The main header is "Jenkins" with a user icon and "Jeff McKenzie | log out". Below the header, there are navigation links "Jenkins" and "All". The main content area has a heading "Enter an item name" with a text input field containing "rudolph". A note below the input says "» Required field". There are three project types listed: "Freestyle project" (selected), "Pipeline", and "multi-configuration project". Each type has a brief description and a small icon. An "OK" button is visible at the bottom left of the list.

Click OK...

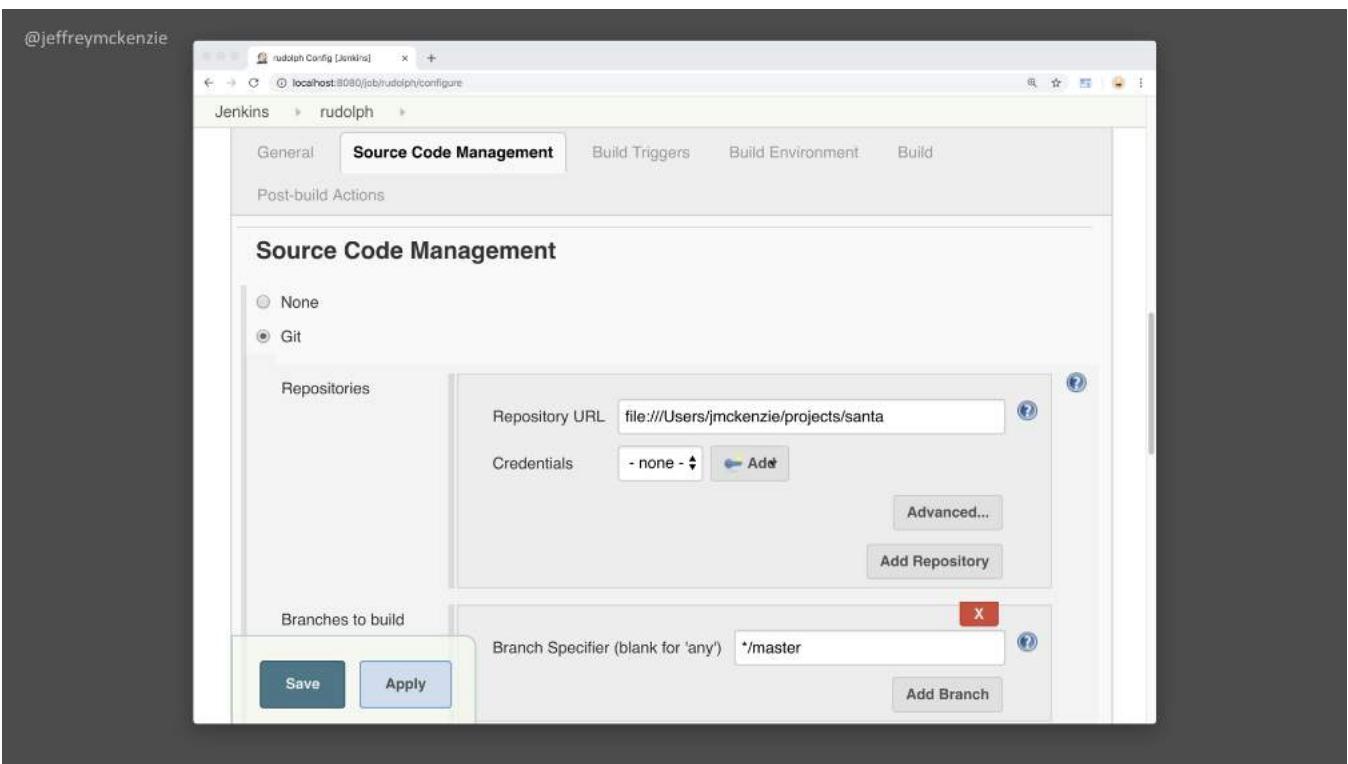


-- description, says hello to Rudolph

-- keep last 3 builds



Under the source code management tab, select Git

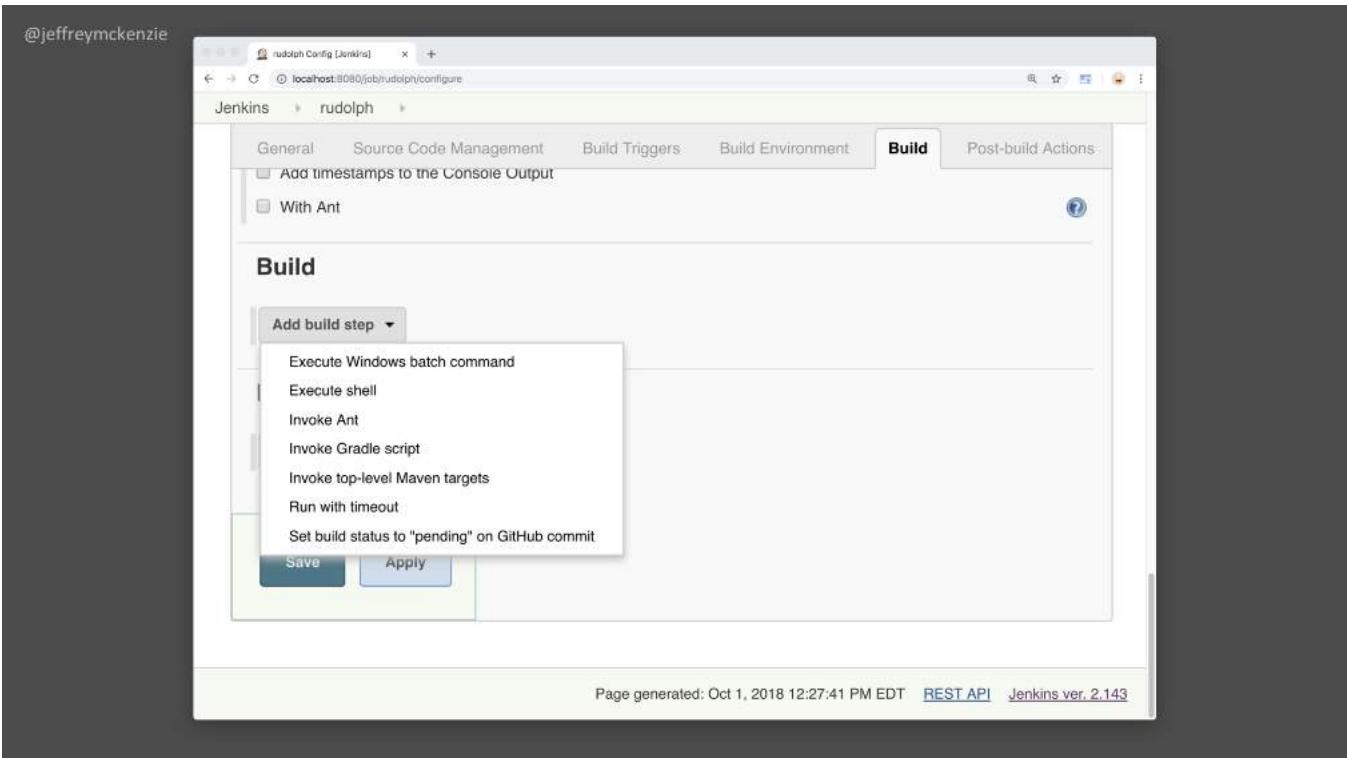


I'm just going to point this to a local git repo

That I have in my home directory

And it's pulling from the master branch

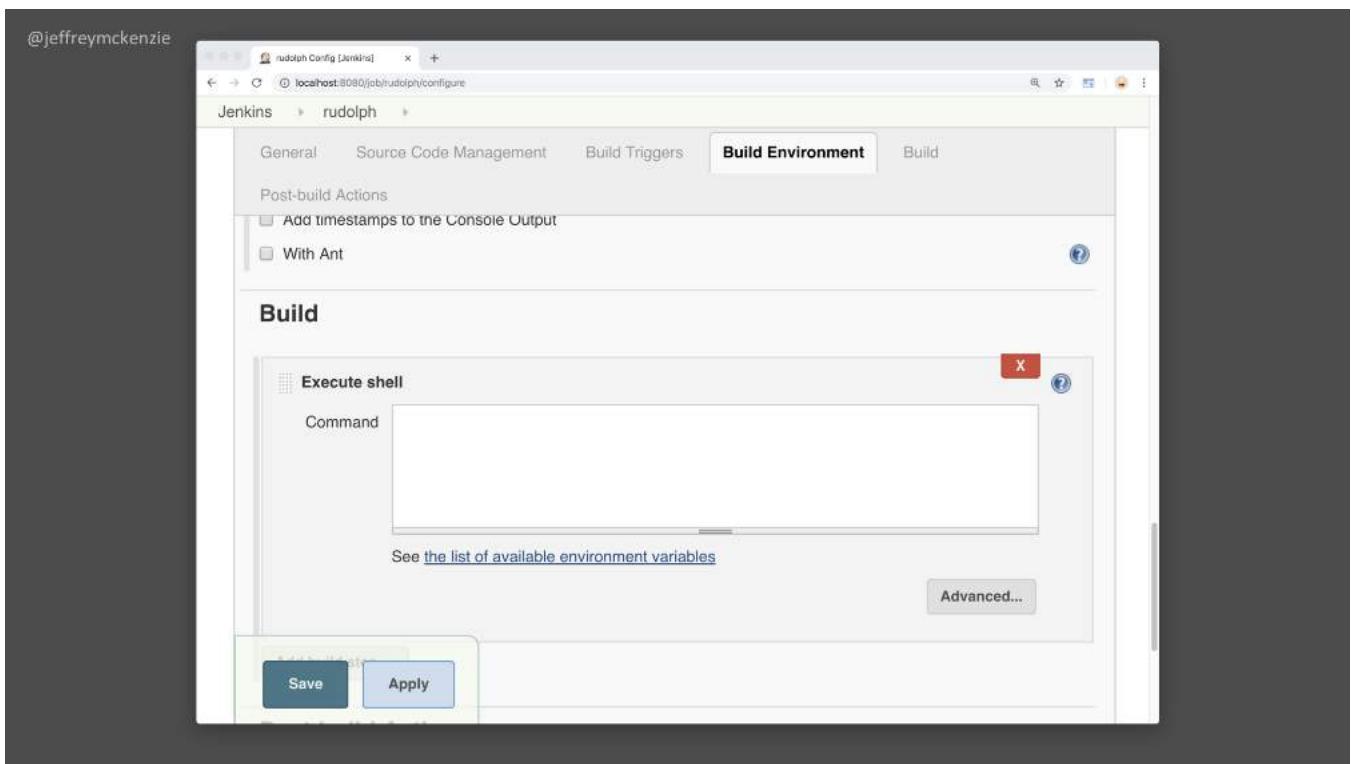
[Git users here?]

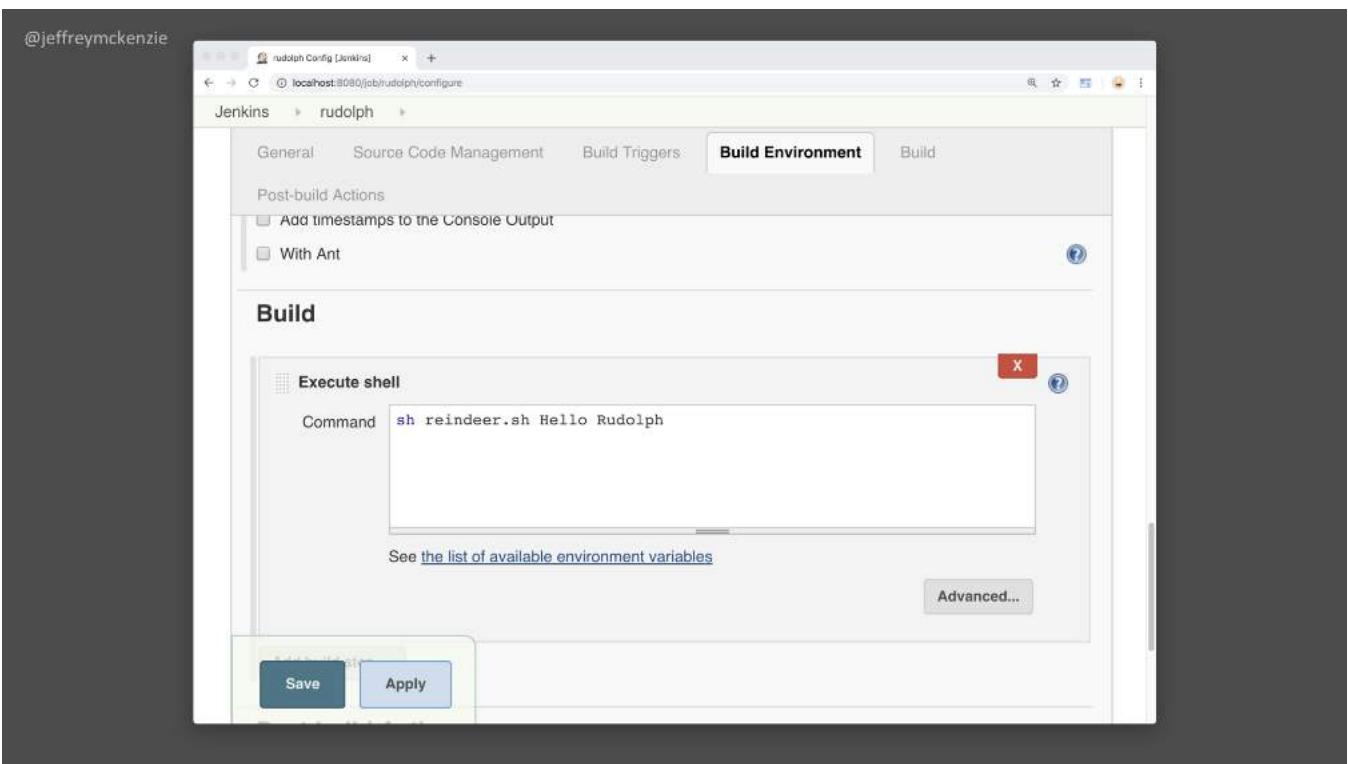


On the build tab, again I'm going to choose

The Execute shell build step...

@jeffreymckenzie





And put the following command in there...

Sh Reindeer.sh Hello Rudolph.

This tells Jenkins to run a shell script called reindeer.sh

That's in my git repo, so let's take a look at that real quick.

@jeffreymckenzie

```
#!/bin/bash
greeting="$1"
reindeer_name="$2"
if [ -z "$greeting" ]
then
    greeting="Hi"
fi
if [ -z "$reindeer_name" ]
then
    reindeer_name="Reindeer"
fi
echo =====
echo =
echo = ${greeting} ${reindeer_name}!
echo =
echo =====
```

So this sets two variables –
Greeting, and reindeer name,
Which get passed in from Jenkins.
If nothing gets passed in for either variable,
Then the script sets defaults.
Then the result gets written as output.

So we will save this, go back to the project screen

@jeffreymckenzie

The screenshot shows the Jenkins interface for the 'rudolph' project. The left sidebar contains links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, and Rename. The main content area is titled 'Project rudolph' with the subtext 'This freestyle job says hello to Rudolph.' It features two buttons: 'edit description' and 'Disable Project'. Below these are sections for 'Workspace' (with a link) and 'Recent Changes' (with a link). A 'Permalinks' section includes a 'Build History' tab (selected), a 'trend' dropdown, a search bar ('find'), and RSS feed links ('RSS for all' and 'RSS for failures'). At the bottom, it says 'Page generated: Oct 1, 2018 1:21:43 PM EDT REST API Jenkins ver. 2.143'.

And click “build Now”

@jeffreymckenzie

The screenshot shows the Jenkins interface for the 'rudolph' project. The left sidebar contains links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, and Rename. The main content area is titled 'Project rudolph' with the sub-header 'This freestyle job says hello to Rudolph.' It features two buttons: 'Edit description' and 'Disable Project'. Below these are sections for 'Workspace' (with a link) and 'Recent Changes' (with a link). A 'Permalinks' section includes a 'Build History' table with one entry: '#1 Oct 1, 2018 1:23 PM'. At the bottom, there are RSS links for 'RSS for all' and 'RSS for failures'. The footer indicates the page was generated on Oct 1, 2018, at 1:21:43 PM EDT, and shows Jenkins version 2.143.

And we have a success – let's look at the output.

```
@jeffreymckenzie
rudolph 21 Jenkins [Jenkins] + Jenkins
localhost:8080/job/rudolph/1/console
Jenkins > rudolph > #1
Console Output

Started by user Jeff McKenzie
Building in workspace /Users/jmckenzie/.jenkins/workspace/rudolph
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url file:///Users/jmckenzie/projects/santa # timeout=10
Fetching upstream changes from file:///Users/jmckenzie/projects/santa
> git --version # timeout=10
> git fetch --tags --progress file:///Users/jmckenzie/projects/santa +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 8dee6bc1eca8126d8b3487eebed8c2de8fc6cce (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 8dee6bc1eca8126d8b3487eebed8c2de8fc6cce
Commit message: "added greeting arg"
First time build. Skipping changelog.
[reindeer] $ /bin/sh -xe /var/folders/35/khrappd13w35brt4j20pgqr4cp_336/T/jenkins4927884320778081077.sh
+ sh reindeer.sh Hello Rudolph
=====
=
= Hello Rudolph!
=
=====

Finished: SUCCESS
```

There we have the result of the script as it's run.

So there's one job for one reindeer.

-- contrived example, but basic

-- does have the basic elements:

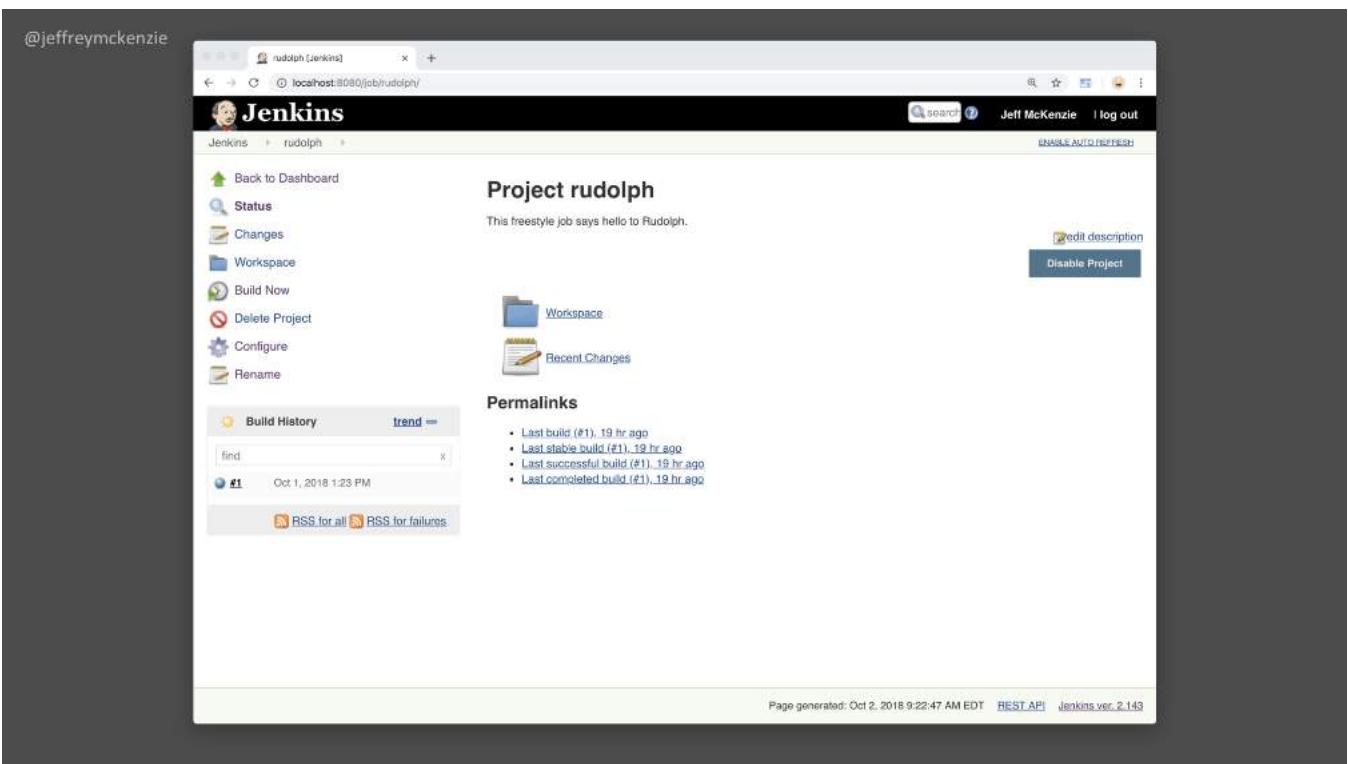
 pulling code from source control

 and doing something with it

--not concerned as much with what it does as

 how to scale and manage these types of jobs

@jeffreymckenzie



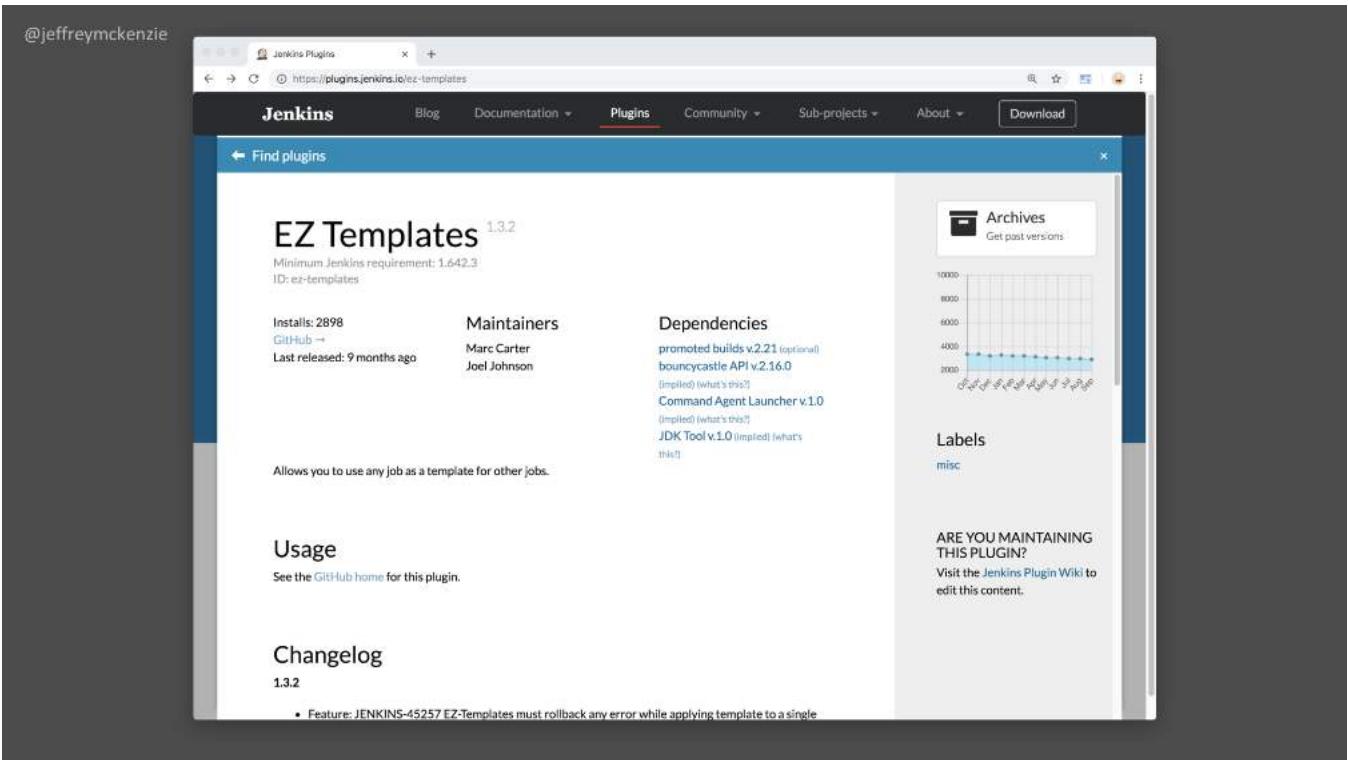
Here's our Rudolph project.

Let's say we copy this job 9 times for a total of 10 reindeer,
And we want this to say something other than Hello as a greeting?

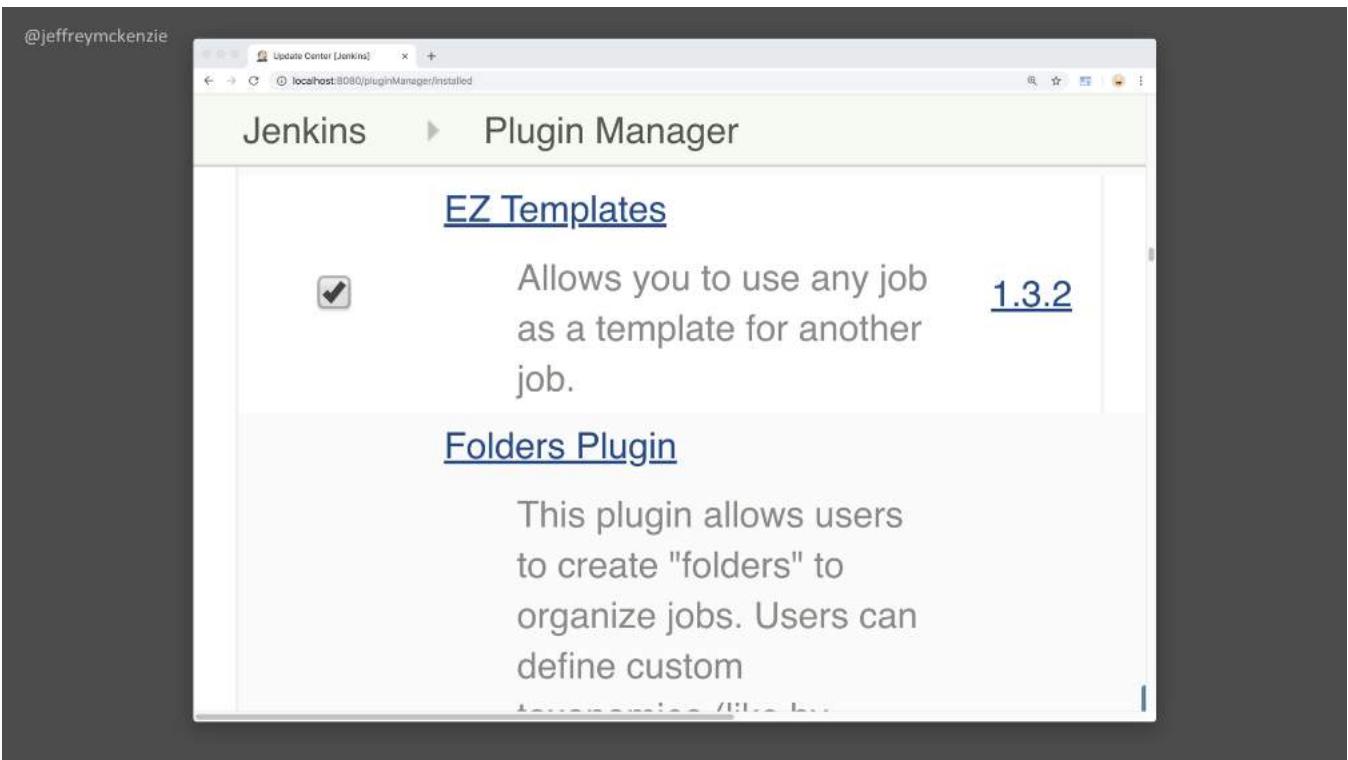
[change each one individually....]

There's a better way to do this....

@jeffreymckenzie



And one way is to use a plugin called EZ Templates
Here's the official plugin page for it,
There's also a git repo for it
the plugin itself is available in the Manage Plugins
section of Jenkins



Like a lot of plugins, this was originally developed by a third party
And then merged into the Jenkins project

So you just need to select and install it

As it describes, this allows you to take any job
And use it as a template for other jobs.

Let's take a look at how it might help us scale in this instance.

@jeffreymckenzie

The screenshot shows the Jenkins dashboard at localhost:8080/view/Jenkins/. The left sidebar contains links for New Item, People, Build History, Edit View, Delete View, Manage Jenkins, My Views, Credentials, and New View. The main area displays a table of Jenkins items:

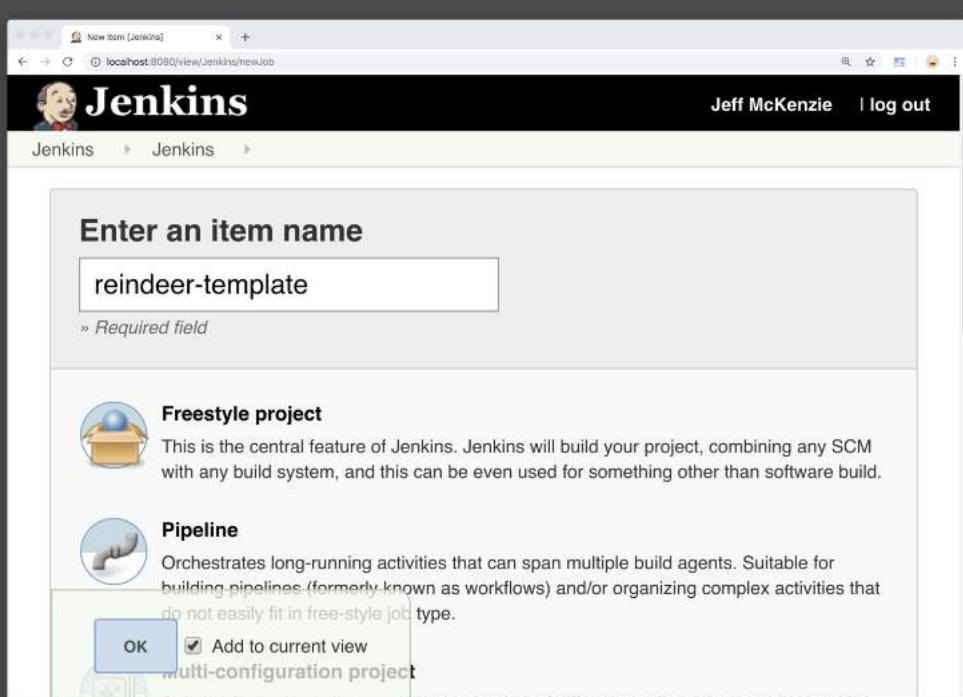
S	W	Name	Last Success	Last Failure	Last Duration
●	○	do-stuff	20 hr - #1	20 hr - #2	49 ms
●	○	rudolph	7 hr 27 min - #1	N/A	0.3 sec

Below the table, there are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). At the bottom right, it says "Page generated: Oct 1, 2018 8:50:27 PM EDT REST API Jenkins ver. 2.143".

So far we have our “do stuff” project,
And our Rudolph project.

Let’s create our template.

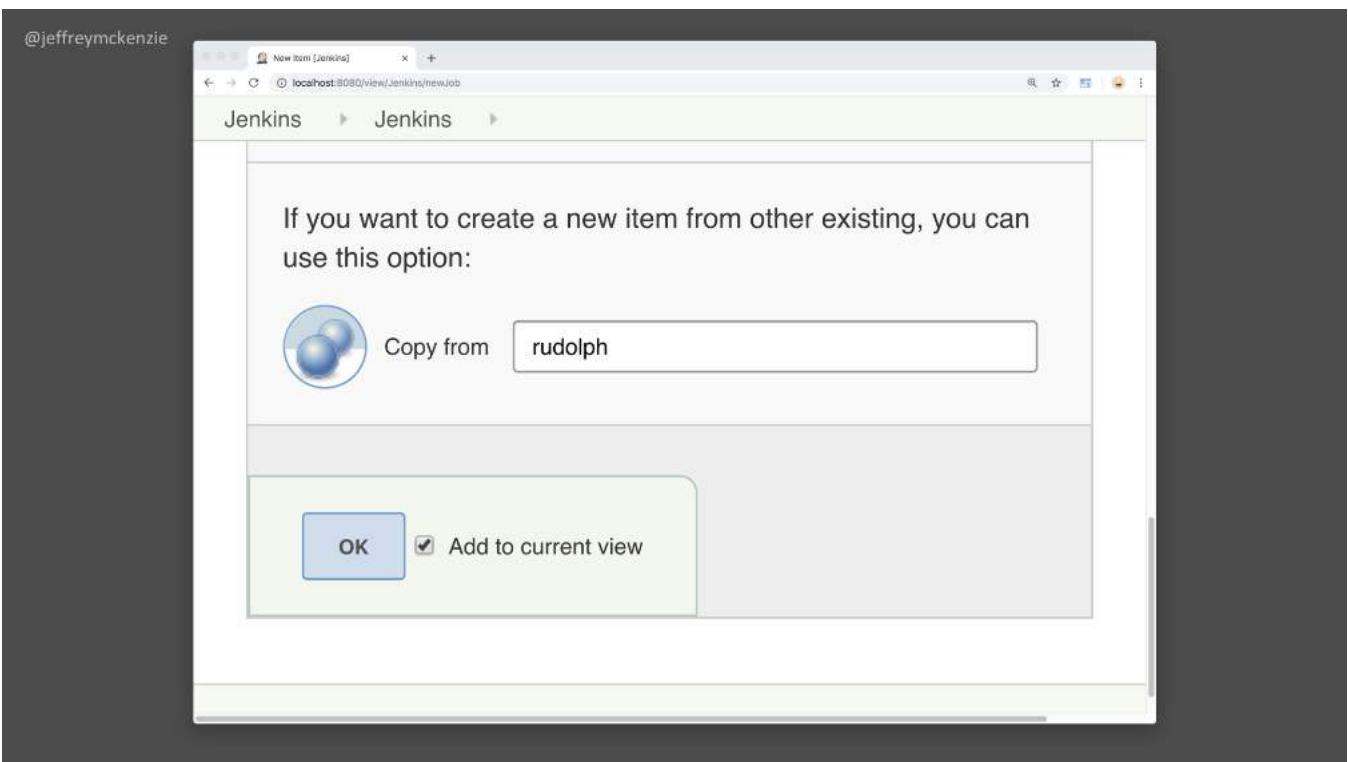
@jeffreymckenzie



The screenshot shows the Jenkins web interface for creating a new item. The title bar says "New Item [Jenkins]". The main header is "Jenkins" with a user icon and "Jeff McKenzie I log out". Below the header, the breadcrumb navigation shows "Jenkins > Jenkins". The main content area has a title "Enter an item name" and a text input field containing "reindeer-template". A note below the input field says "» Required field". Below this, there are two project types listed: "Freestyle project" and "Pipeline". The "Freestyle project" section includes a description: "This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build." The "Pipeline" section includes a description: "Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type." At the bottom of the form are two buttons: "OK" and "Cancel". There is also a checked checkbox labeled "Add to current view".

Let's call it Reindeer Template...

@jeffreymckenzie



And we will create it as a copy of Rudolph

So we can use that as a starting point.

@jeffreymckenzie

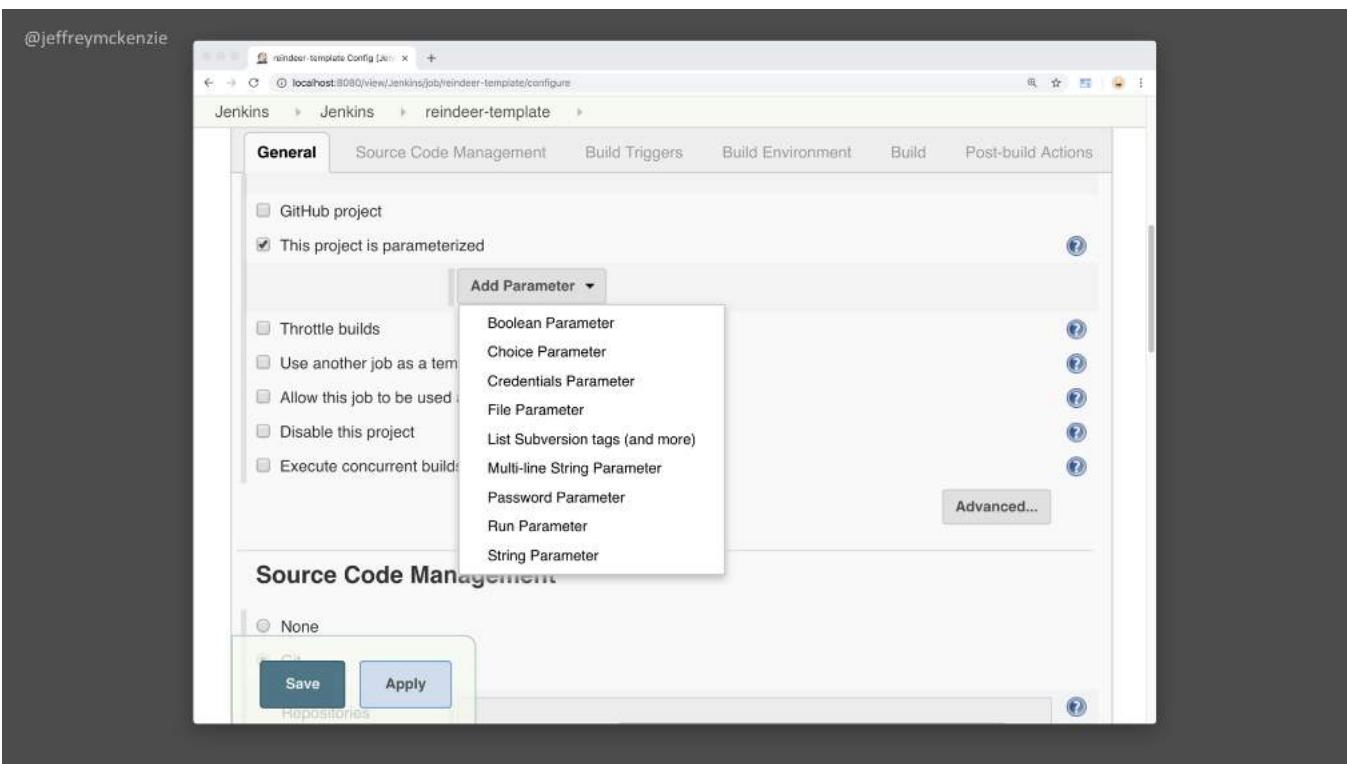
The screenshot shows the Jenkins configuration interface for a job named 'reindeer-template'. The top navigation bar includes the Jenkins logo, a search bar, and the user 'Jeff McKenzie | log out'. The main content area has tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is selected. Under 'Description', there is a text input field containing the text: 'This freestyle job says hello to Rudolph.' Below this is a link '[Plain text] Preview'. A checked checkbox labeled 'Discard old builds' is present. The 'Strategy' section contains a dropdown menu set to 'Log Rotation' and a 'Days to keep builds' input field with the value '3'. A note below it states: 'if not empty, build records are only kept up to this number of days'. Another input field 'Max # of builds to keep' is set to '3', with a note: 'if not empty, only up to this number of build records are kept'. At the bottom left are 'Save' and 'Apply' buttons.

Let's go ahead and change the description
So it makes sense as a template

@jeffreymckenzie

The screenshot shows the Jenkins configuration interface for a job named 'reindeer-template'. The top navigation bar includes the Jenkins logo, a search bar, and the user 'Jeff McKenzie | log out'. Below the header, the breadcrumb navigation shows 'Jenkins > Jenkins > reindeer-template'. The main content area has tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is selected. In the 'Description' field, the text 'Serves as a template for the Reindeer jobs.' is entered. Below this, there is a link '[Plain text] Preview'. A checked checkbox labeled 'Discard old builds' is present. Under the 'Strategy' section, a dropdown menu is set to 'Log Rotation'. A 'Days to keep builds' input field contains the value '3', with a note below stating 'if not empty, build records are only kept up to this number of days'. Another input field for 'Max # of builds to keep' also contains the value '3', with a note below stating 'if not empty, only up to this number of build records are kept'. At the bottom left are 'Save' and 'Apply' buttons.

“Serves as a template for the reindeer jobs”

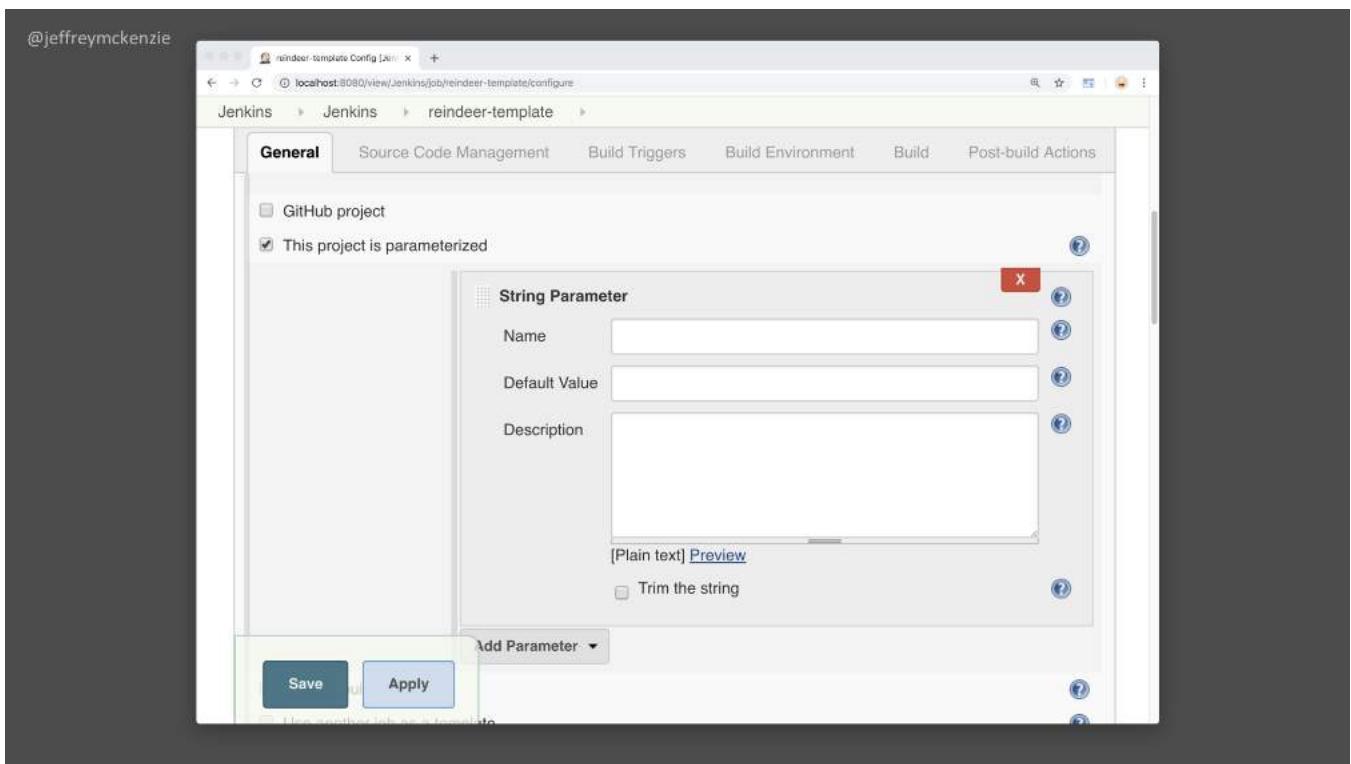


We are also going to add a couple of parameters (variables)
So Santa can personalize his reindeer greeting –

Check box for “project is parameterized”

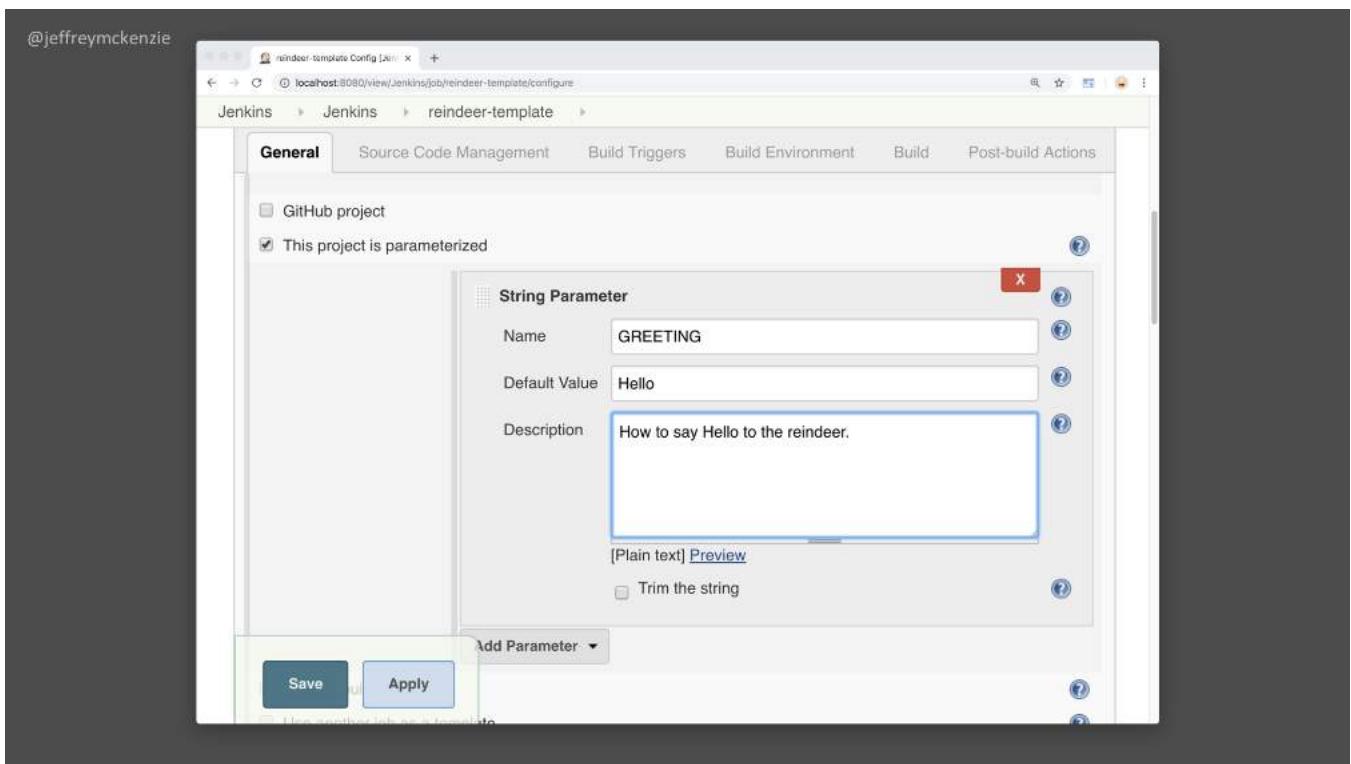
And we will select the string parameter

@jeffreymckenzie



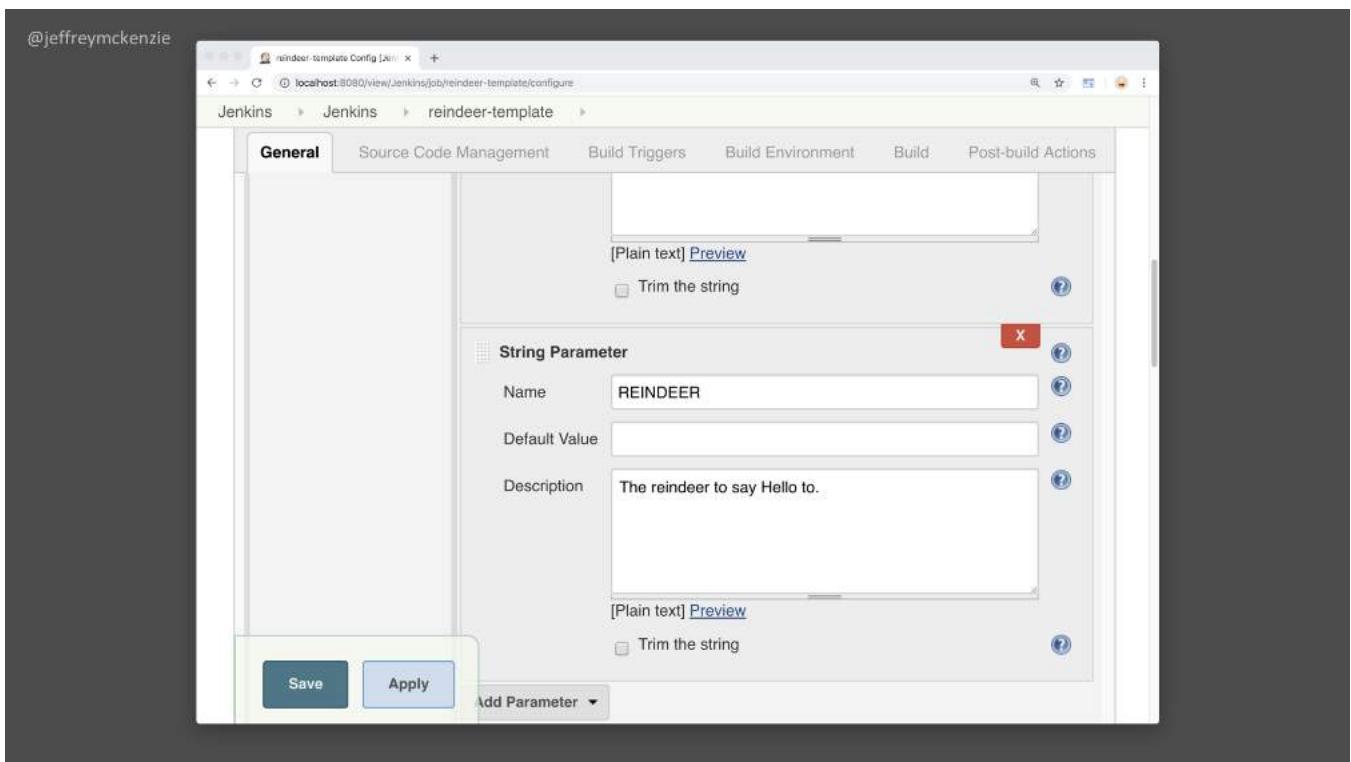
Let's enter our values...

@jeffreymckenzie

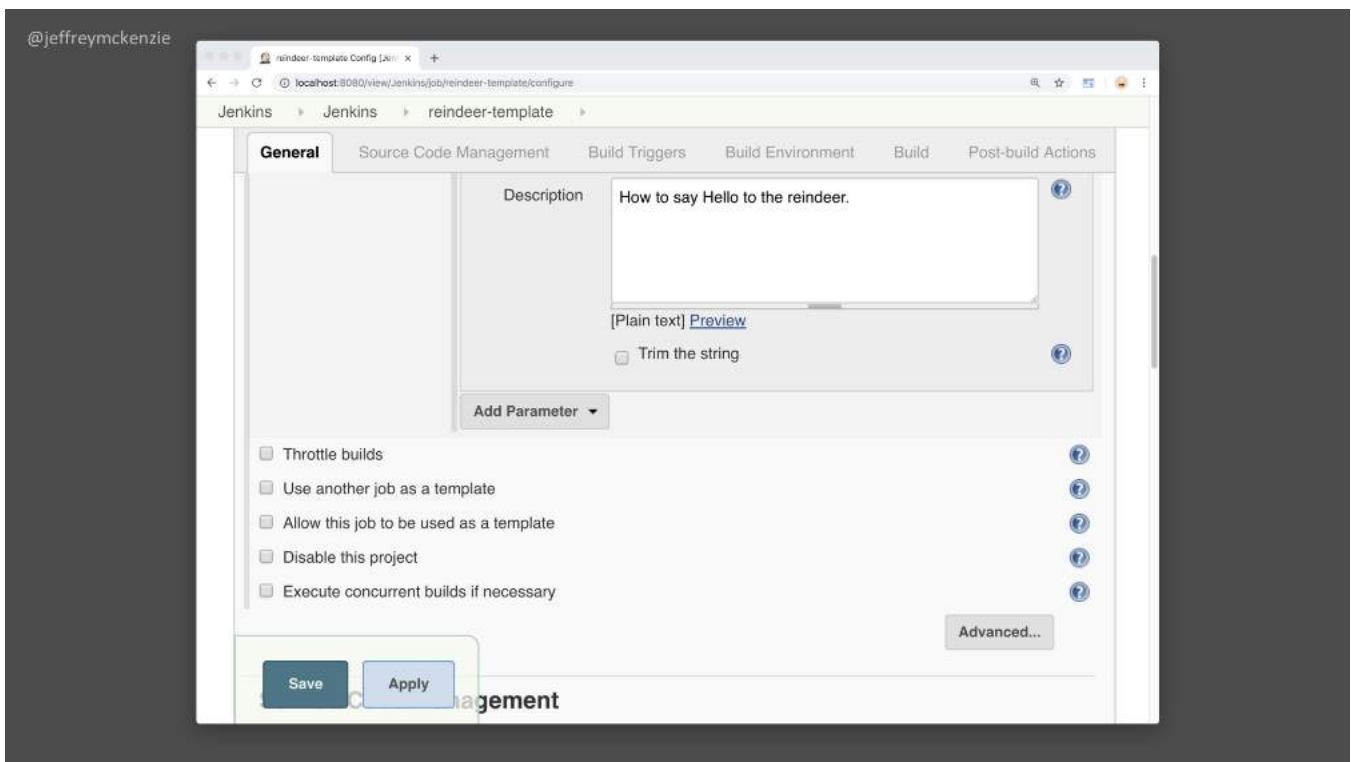


We'll call this parameter GREETING
With a default value of Hello
And also give it a description.

@jeffreymckenzie

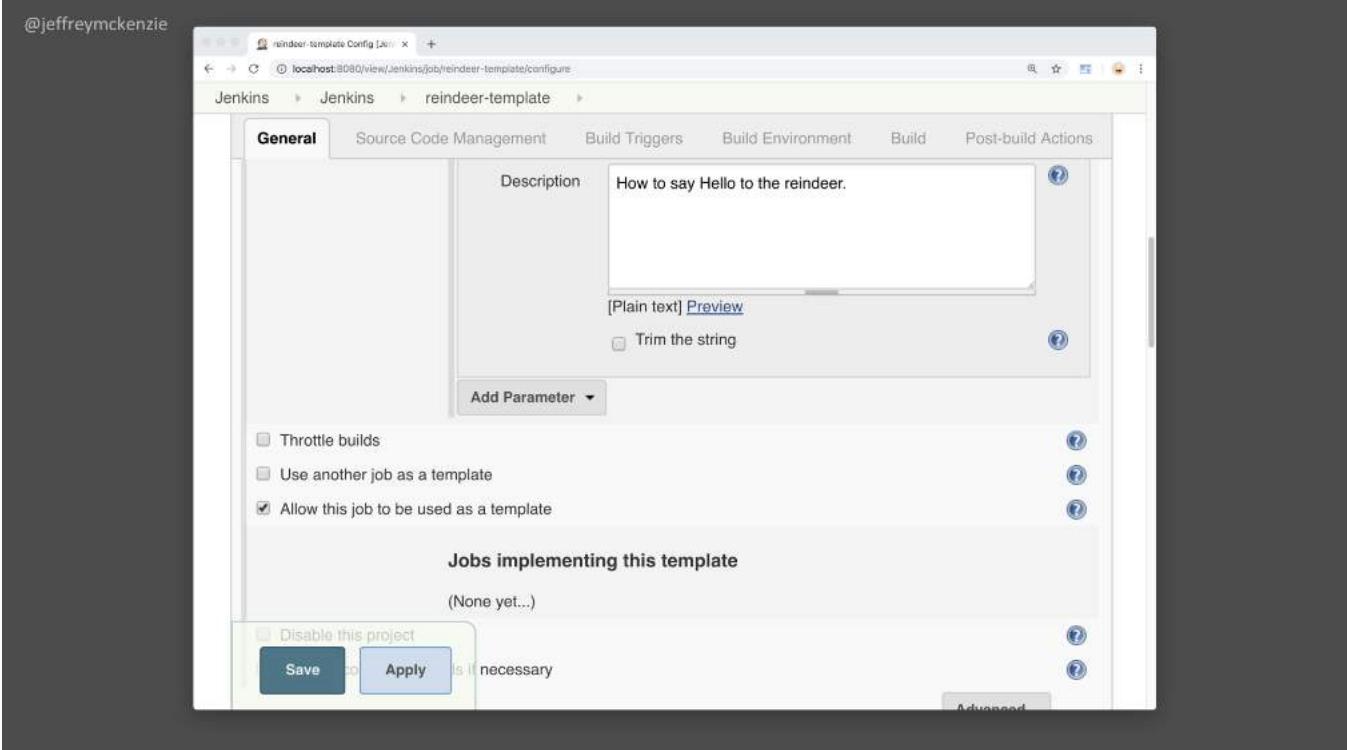


Next we will add parameter REINDEER
With no default value
And also give it a description –
“The reindeer to say hello to YOUR MOM.”

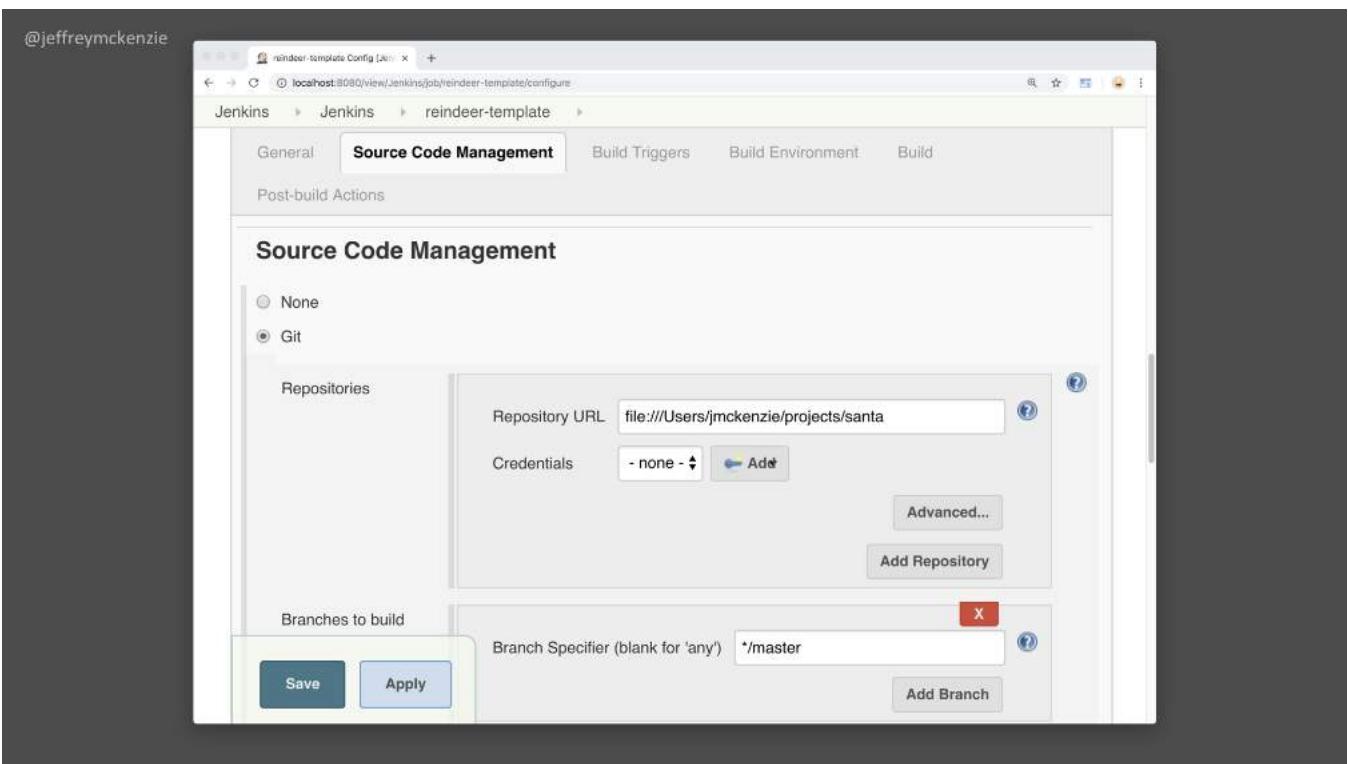


Next we will scroll down to the checkbox that says
“Allow this job to be used as a template”

[you see this option only if you have EZ Templates installed]

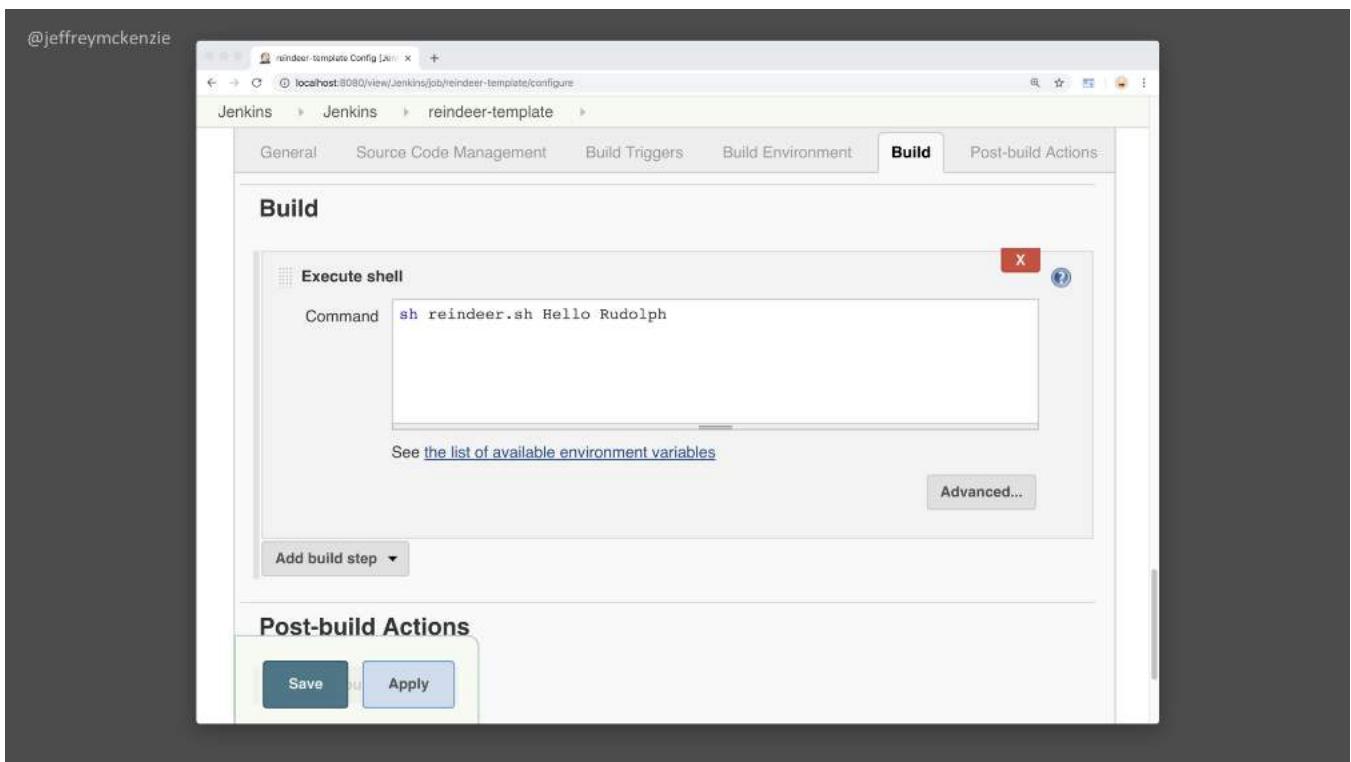


When you check the box, you see a list
Of jobs using the template
(none yet since we just created it)



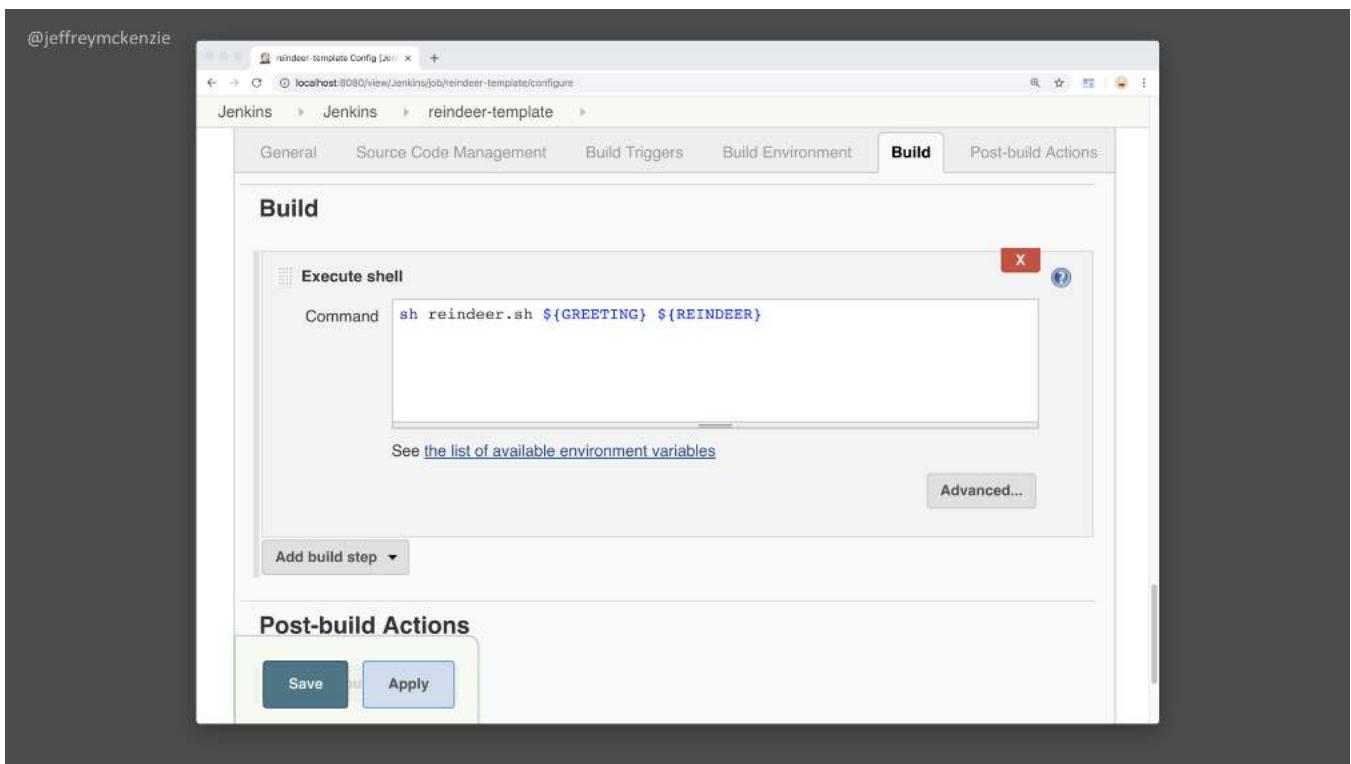
We leave our git repo as it is....

@jeffreymckenzie



We will also change our shell command
To use the parameters

@jeffreymckenzie



Then we will click save...

@jeffreymckenzie

The screenshot shows a Jenkins project page titled "Project reindeer-template". The page header includes the Jenkins logo, a search bar, and user information for "Jeff McKenzie". A sidebar on the left contains links such as "Back to Dashboard", "Status", "Changes", "Workspace", "Build with Parameters", "Delete Project", "Configure", and "Rename". The main content area displays the project's purpose ("Serves as a template for the Reindeer jobs."), a note about being a template job ("ez-templates: This job is configured as a template"), and links for "Workspace" and "Recent Changes". Below this is a "Permalinks" section with a search bar and RSS feed links. The footer of the page provides generation details ("Page generated: Oct 2, 2018 11:21:28 AM EDT") and links to "REST API" and "Jenkins ver. 2.143".

Which takes us back to the project page

@jeffreymckenzie

The screenshot shows a Jenkins project page for 'reindeer-template'. The title bar says 'reindeer-template [Jenkins]'. The URL is 'localhost:8080/view/Jenkins/job/reindeer-template/'. The top navigation bar has 'Jenkins' and 'reindeer-template' tabs, and an 'ENABLE AUTO REFRESH' button. Below the title, it says 'Project reindeer-template' and 'Serves as a template for the Reindeer jobs.' There is a green 'ez-templates' icon with the text 'ez-templates: This job is configured as a template'. On the right, there is an 'edit description' link and a 'Disable Project' button. Below these are links for 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). A section titled 'Permalinks' is shown. At the bottom, a footer bar includes 'Page generated: Oct 2, 2018 11:21:28 AM EDT', 'REST API', and 'Jenkins ver. 2.143'.

If we look closer you'll see there's
A note that this project is configured as a template.

Because this is a template and not something we want to run,
We will disable the project, which is important
If the template contains build triggers
Or anything like that.

@jeffreymckenzie

The screenshot shows a Jenkins job configuration page. The title bar says "reindeer-template [Jenkins]". The URL in the address bar is "localhost:8080/view/Jenkins/job/reindeer-template/". The page header includes "Jenkins" twice, a "ENABLE AUTO REFRESH" link, and a search icon. Below the header, the main content area has a heading "Project reindeer-template" and a sub-heading "Serves as a template for the Reindeer jobs.". A green puzzle piece icon indicates it's a template. A yellow warning icon says "This project is currently disabled" next to an "Enable" button. There are links for "Workspace" (with a folder icon) and "Recent Changes" (with a notebook icon). At the bottom, there's a "Permalinks" section and a footer with "Page generated: Oct 2, 2018 11:24:20 AM EDT REST API Jenkins ver. 2.143".

So we will do that
And it shows up as disabled.

@jeffreymckenzie

The screenshot shows the Jenkins dashboard at localhost:8080/view/Jenkins/. The left sidebar includes links for New Item, People, Build History, Edit View, Delete View, Manage Jenkins, My Views, Credentials, and New View. The main content area displays a table of projects under the 'Jenkins' tab. The table has columns for S (Status), W (Last build), Name, Last Success, Last Failure, and Last Duration. The projects listed are:

S	W	Name	Last Success	Last Failure	Last Duration
Red	do-stuff	do-stuff	1 day 11 hr - #1	1 day 11 hr - #2	49 ms
Grey	Yellow	reindeer-template	N/A	N/A	N/A
Blue	Yellow	rudolph	22 hr - #1	N/A	0.3 sec

Below the table, there are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). At the bottom right, it says "Page generated: Oct 2, 2018 11:25:27 AM EDT REST API Jenkins ver. 2.143".

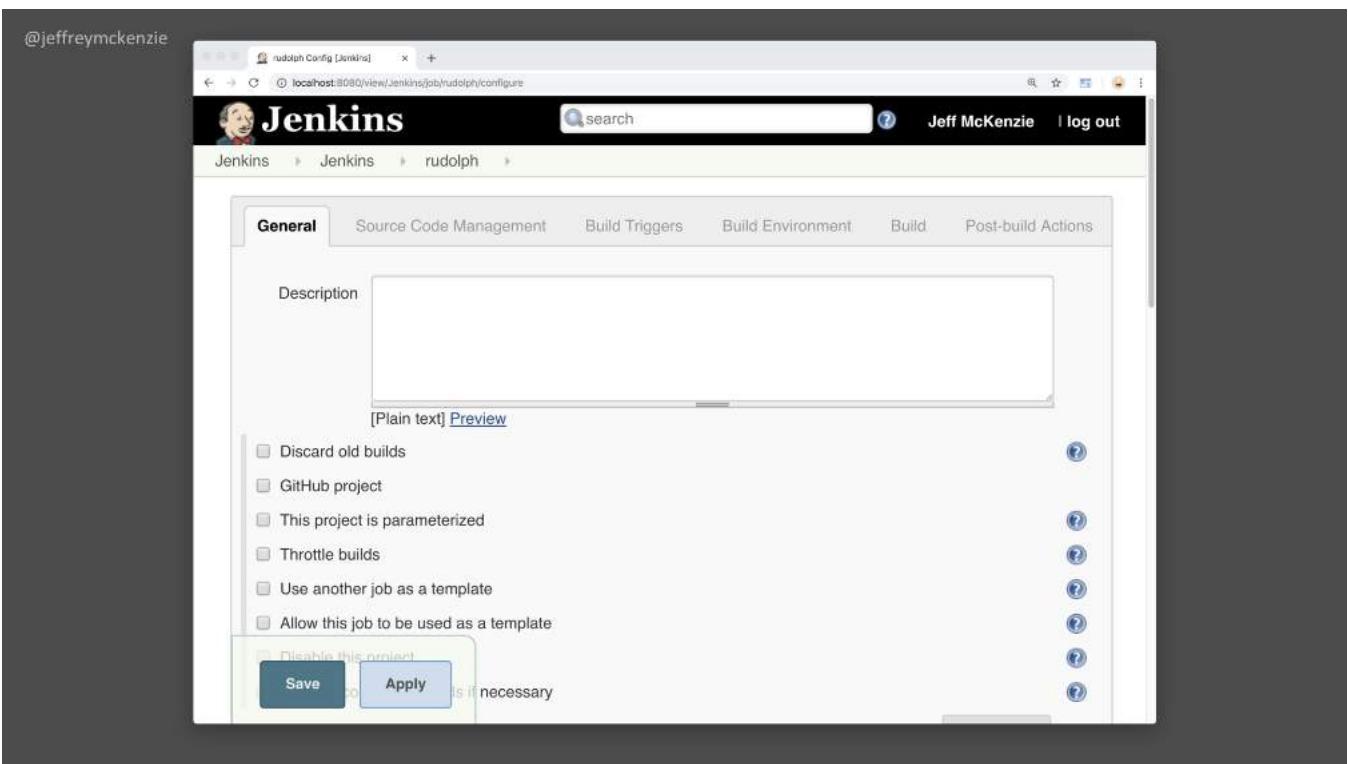
Now we have three projects: our `dostuff` example,
Our Rudolph project,
And our template.

Lets delete and recreate our Rudolph project to use the template.

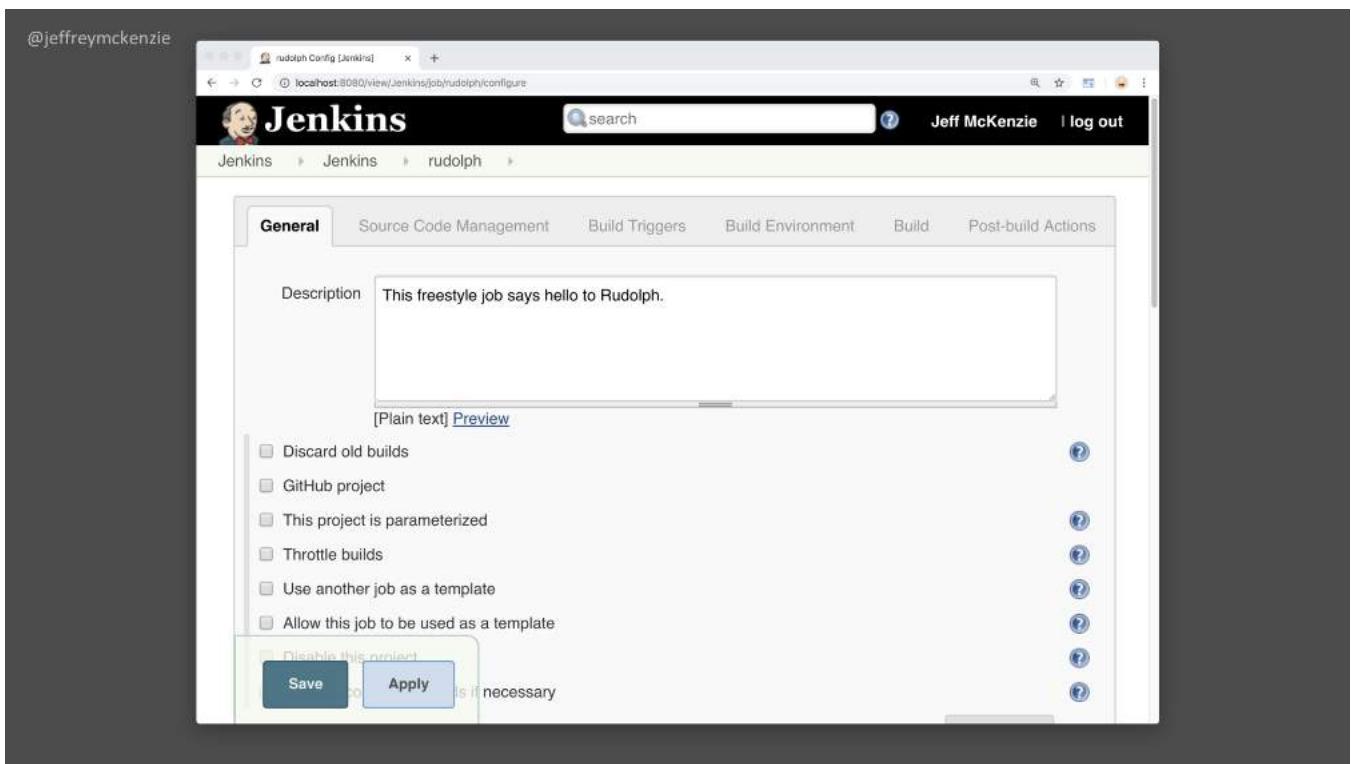
@jeffreymckenzie

The screenshot shows the Jenkins dashboard at localhost:8080/view/Jenkins/. The left sidebar contains links for New Item, People, Build History, Edit View, Delete View, Manage Jenkins, My Views, Credentials, and New View. The main area displays a table of jobs: 'do-stuff' (last success: 1 day 11 hr - 111, last failure: 1 day 11 hr - 112, duration: 49 ms) and 'reindeer-template' (N/A). A legend indicates icons for S (Stable), W (Warning), and L (Last Failure). Below the table are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The bottom right corner shows page generation details: Page generated: Oct 2, 2018 11:33:12 AM EDT, REST API, Jenkins ver. 2.144.

Ok Rudolph is gone,
Let's create a new freestyle job
Called Rudolph and get into it.



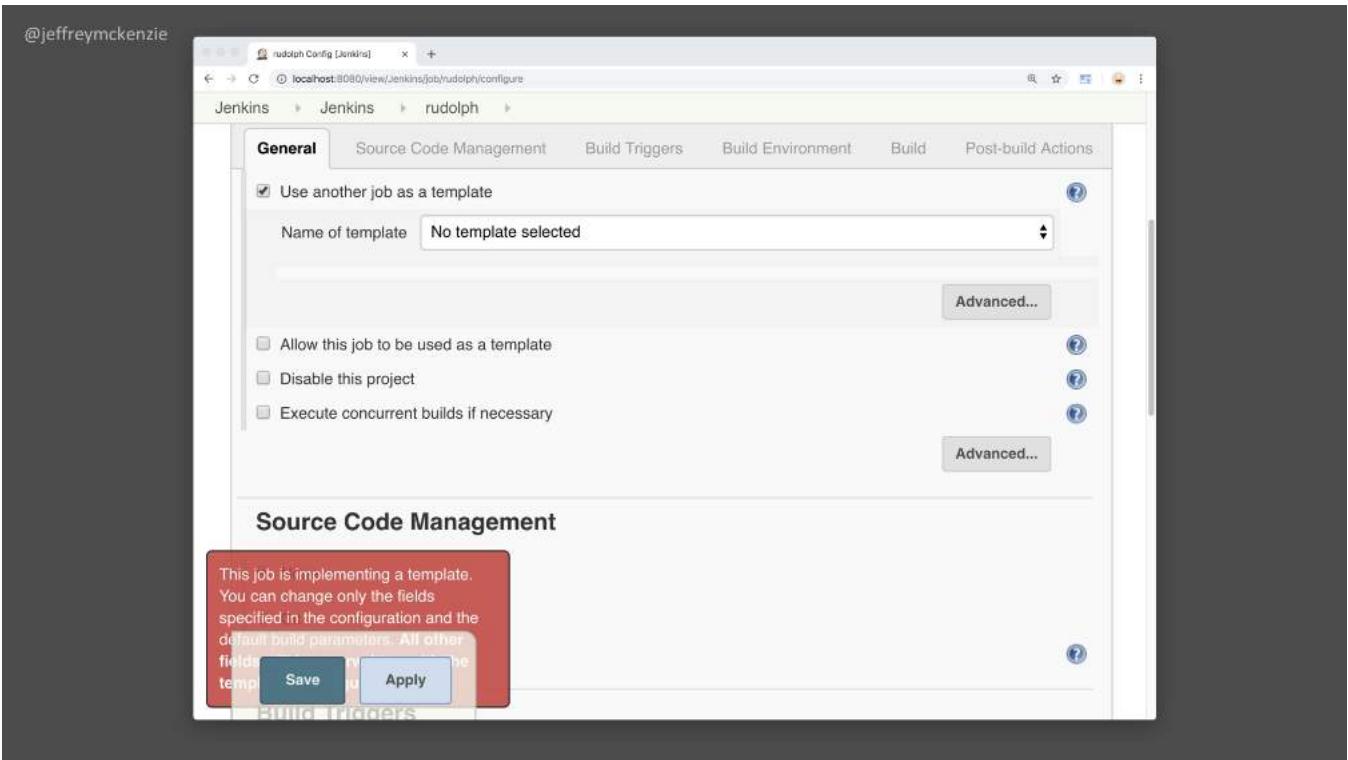
We'll use the same description we had before....



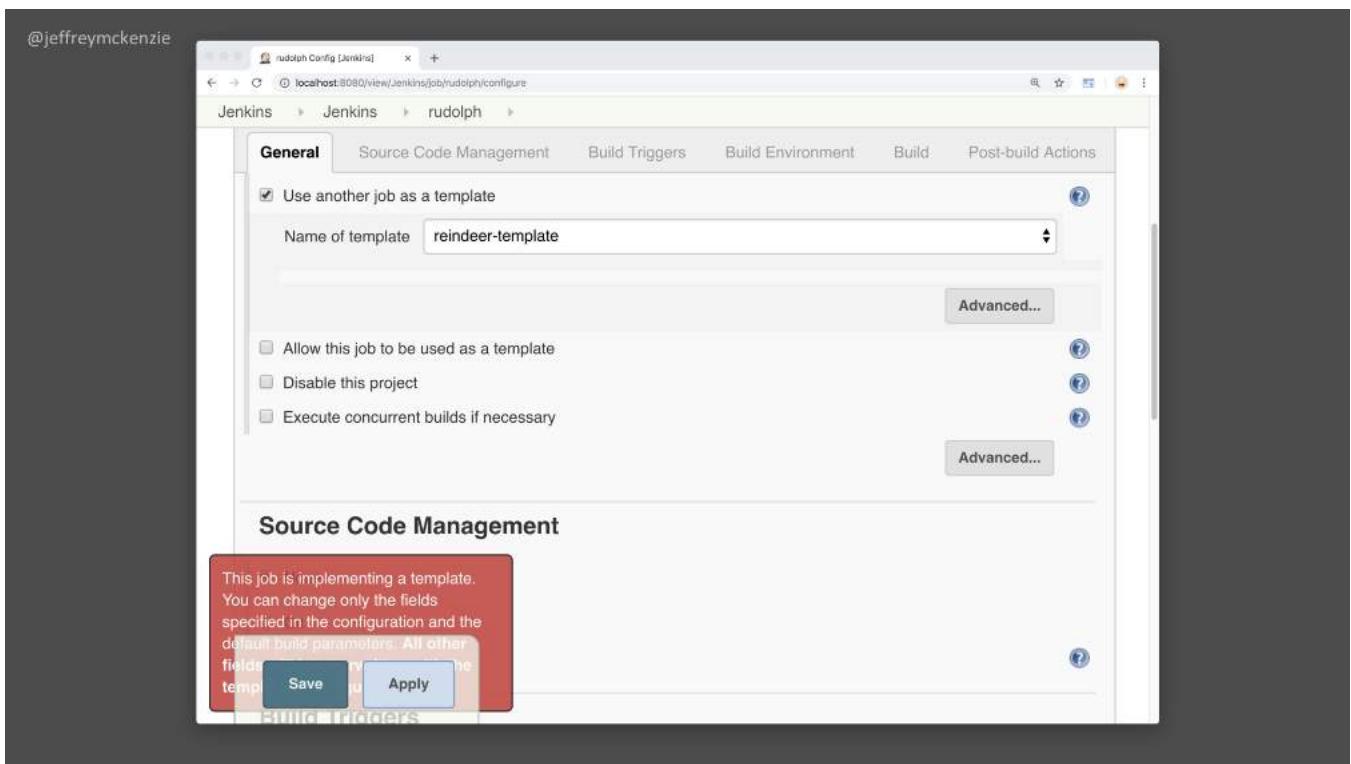
This freestyle job says hello to Rudolph....

Then we scroll down to the checkbox that says

“use another job as a template”

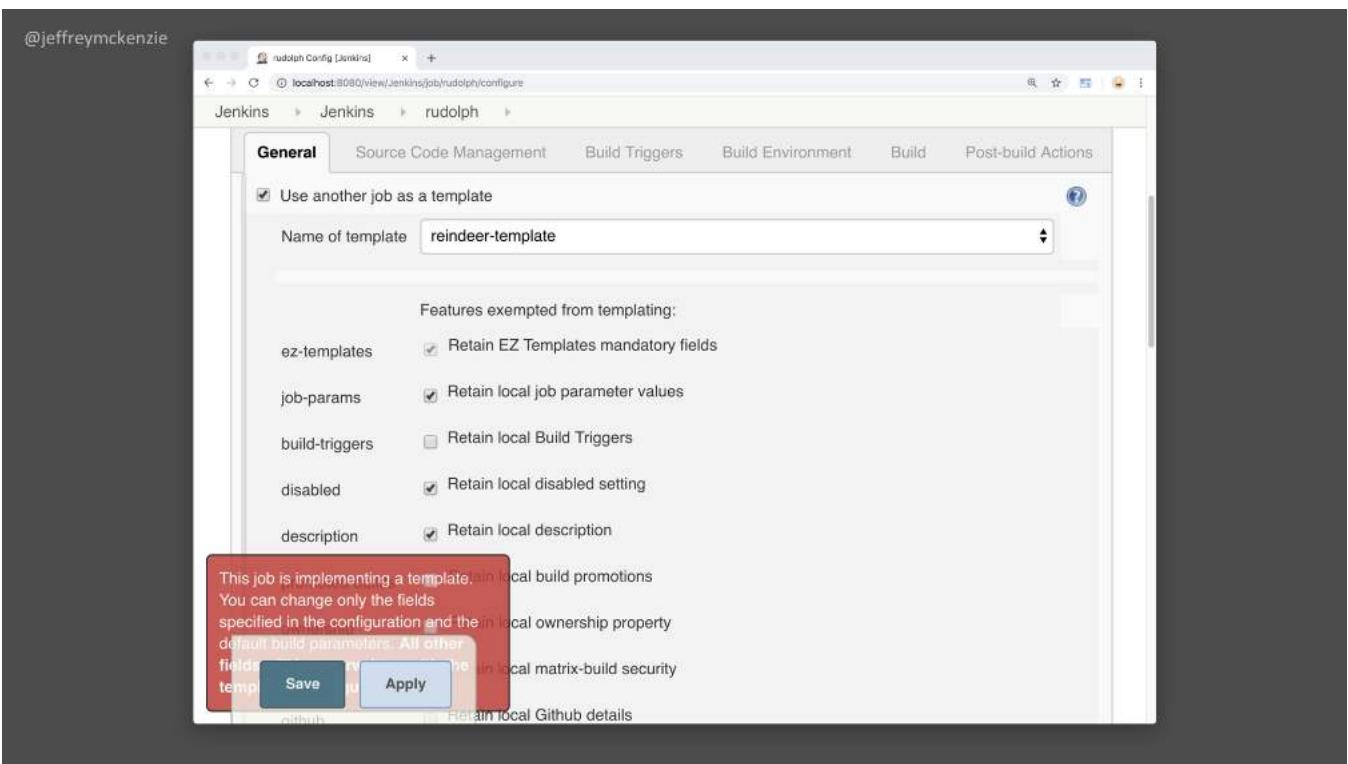


When we check that we are given a dropdown
Of what template we want to select



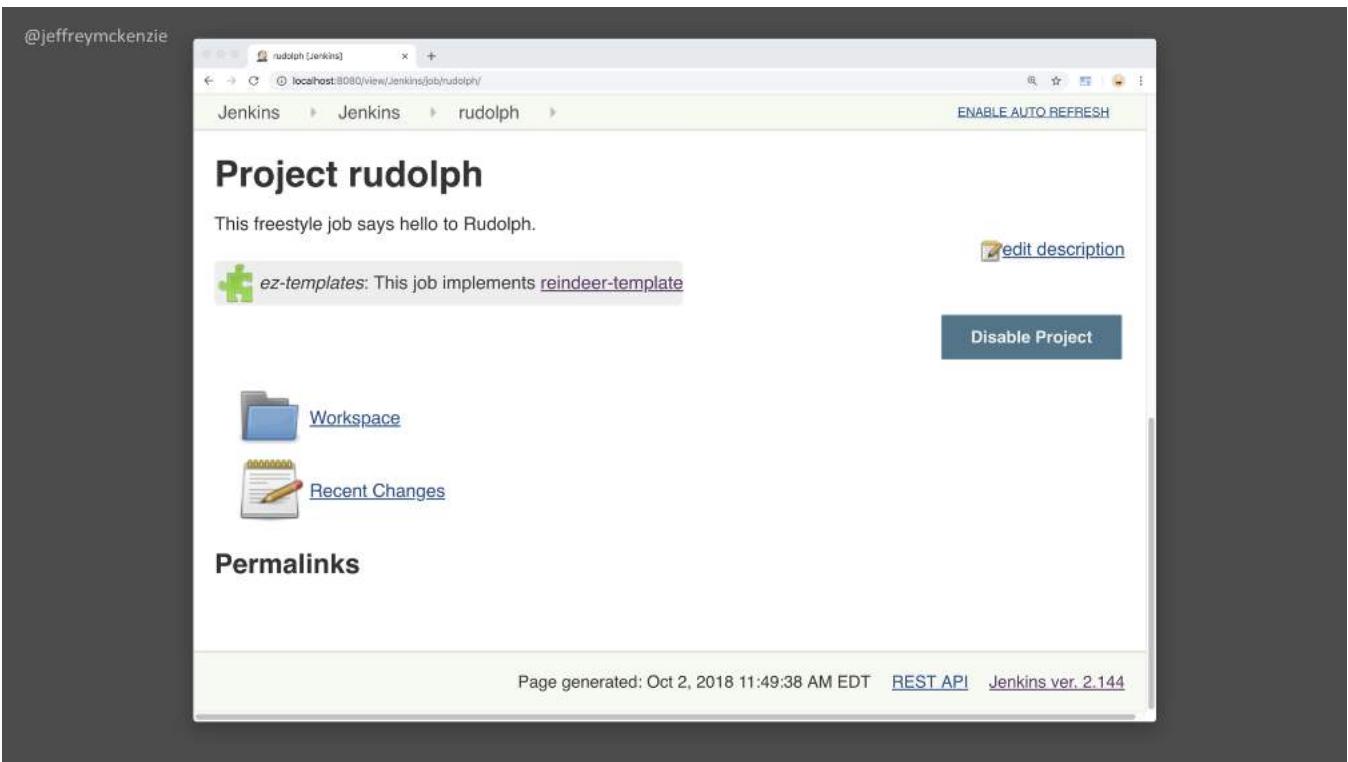
We will choose reindeer template

If you click the Advanced button on the right side
You will get a bunch of additional options
On how you want this job to interact



Such as “retain local job parameter values”
Which we will see in a minute.

Let’s save that and go back to the project page

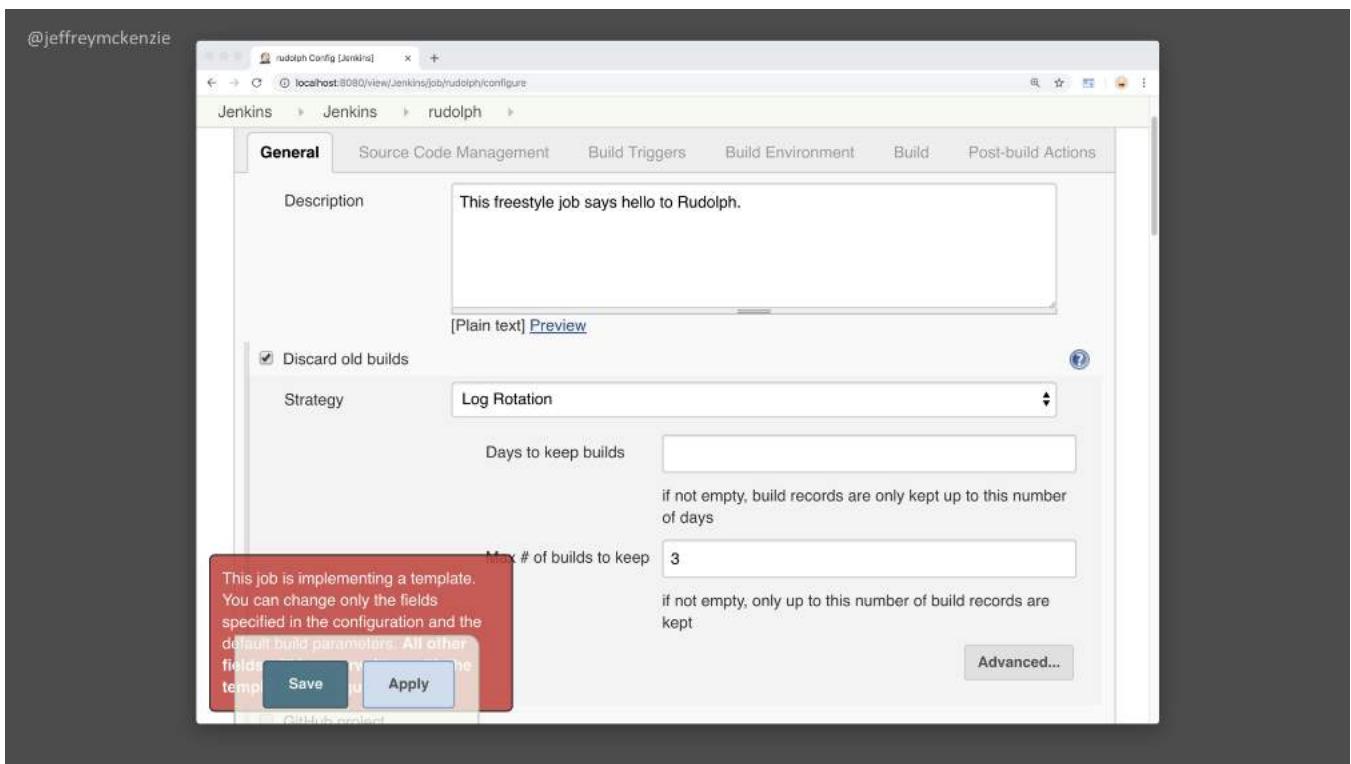


This shows us that Rudolph
Is implementing the reindeer template.

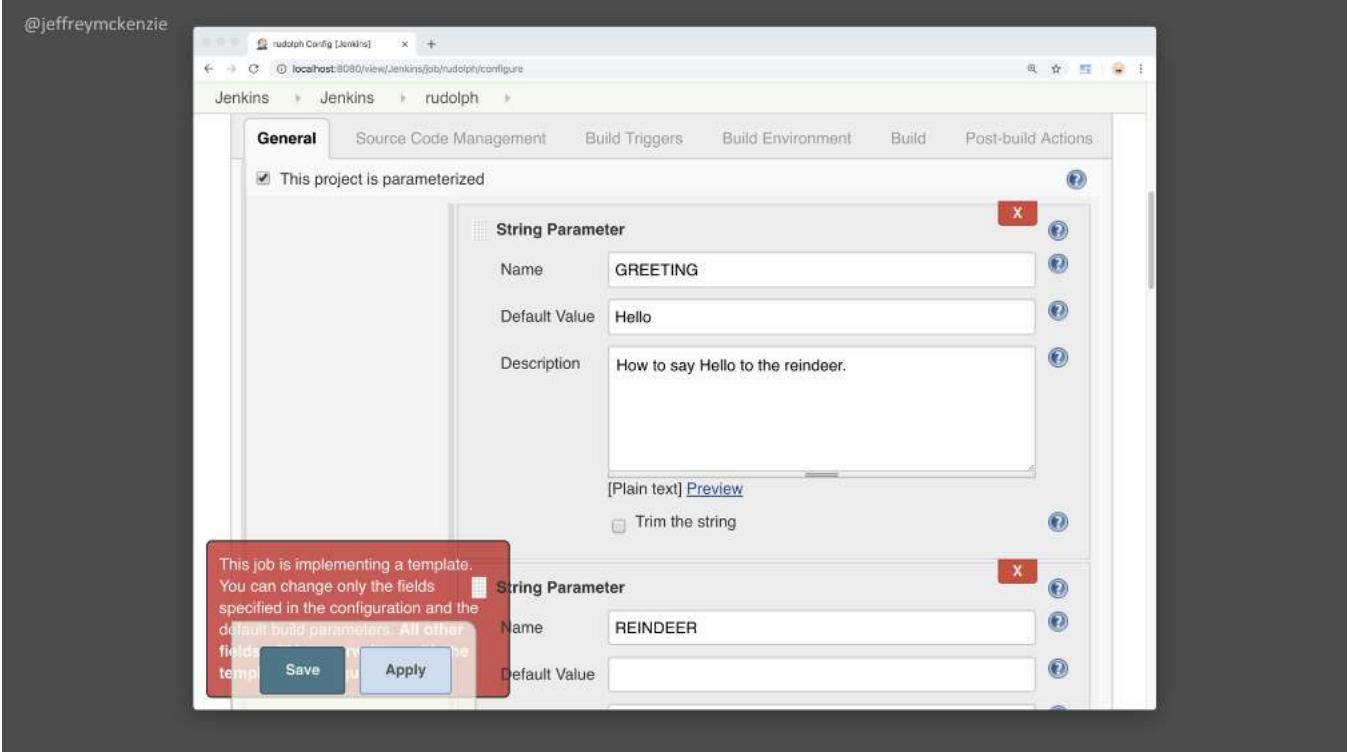
Remember, all we did was add a description,
Select a template, and save it.

Let's go back into the configuration.

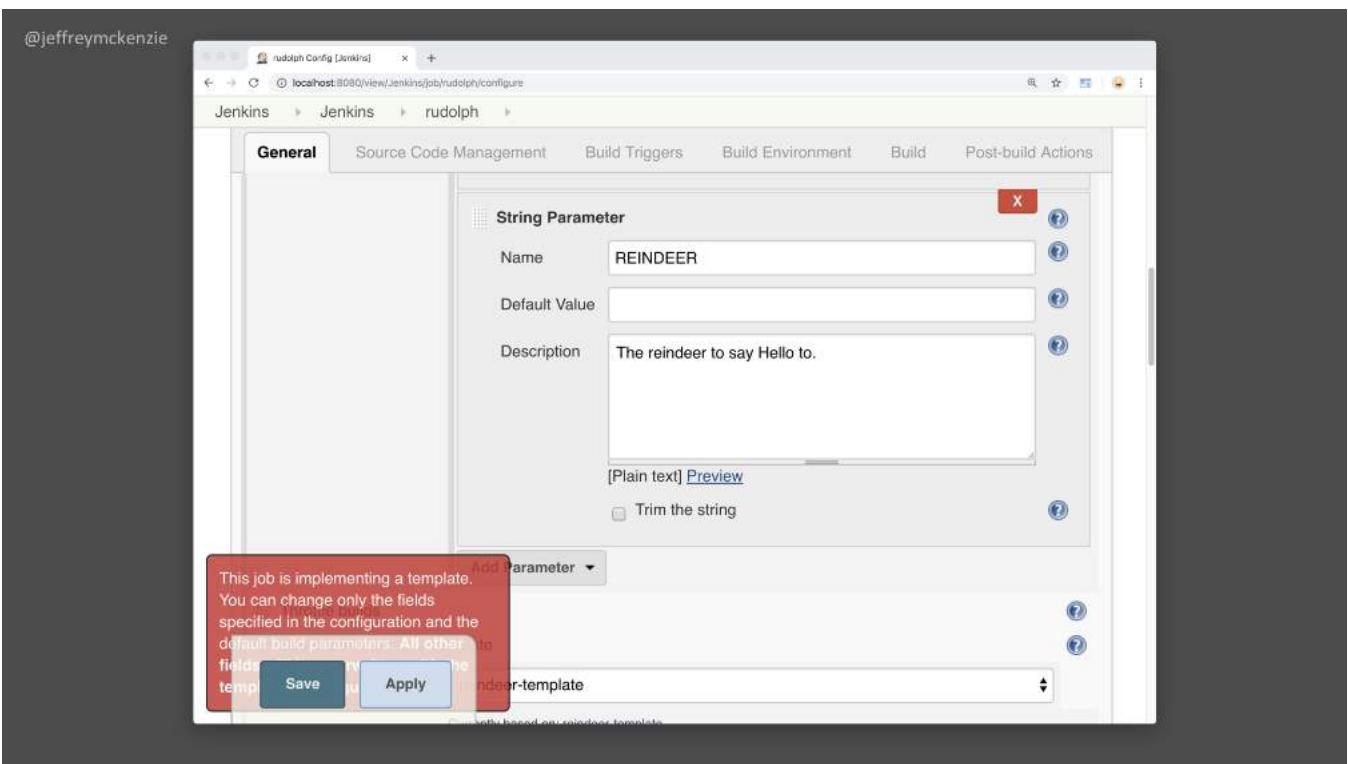
@jeffreymckenzie



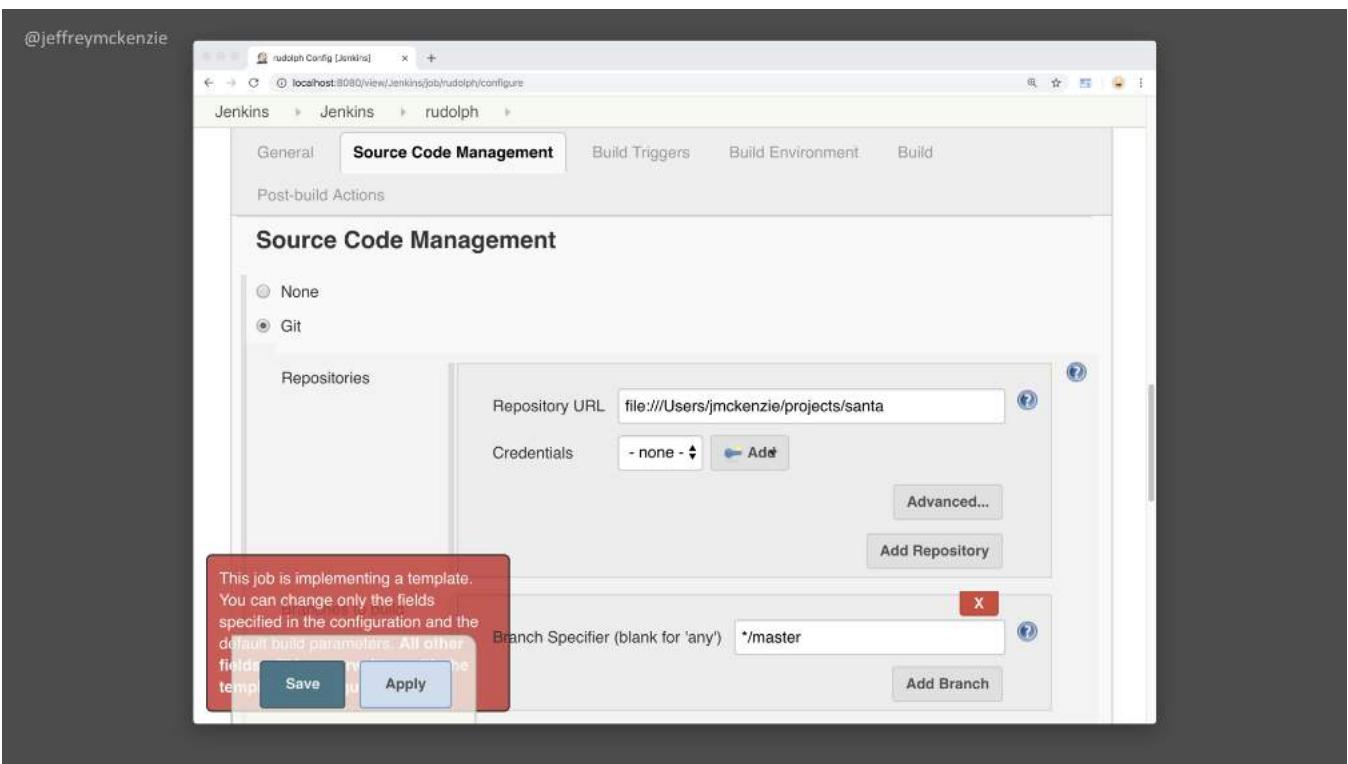
We have our description,
Our build history setting...



Our GREETING parameter....

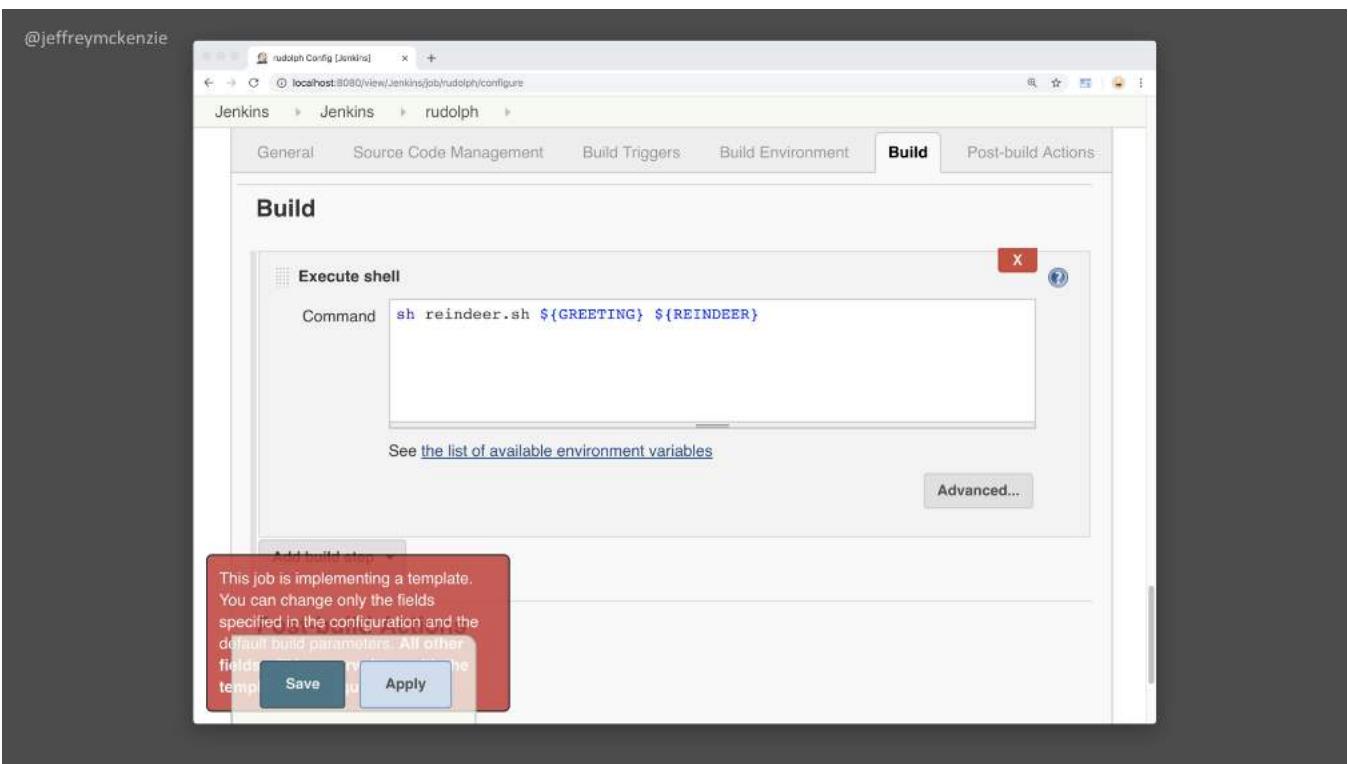


Our REINDEER parameter



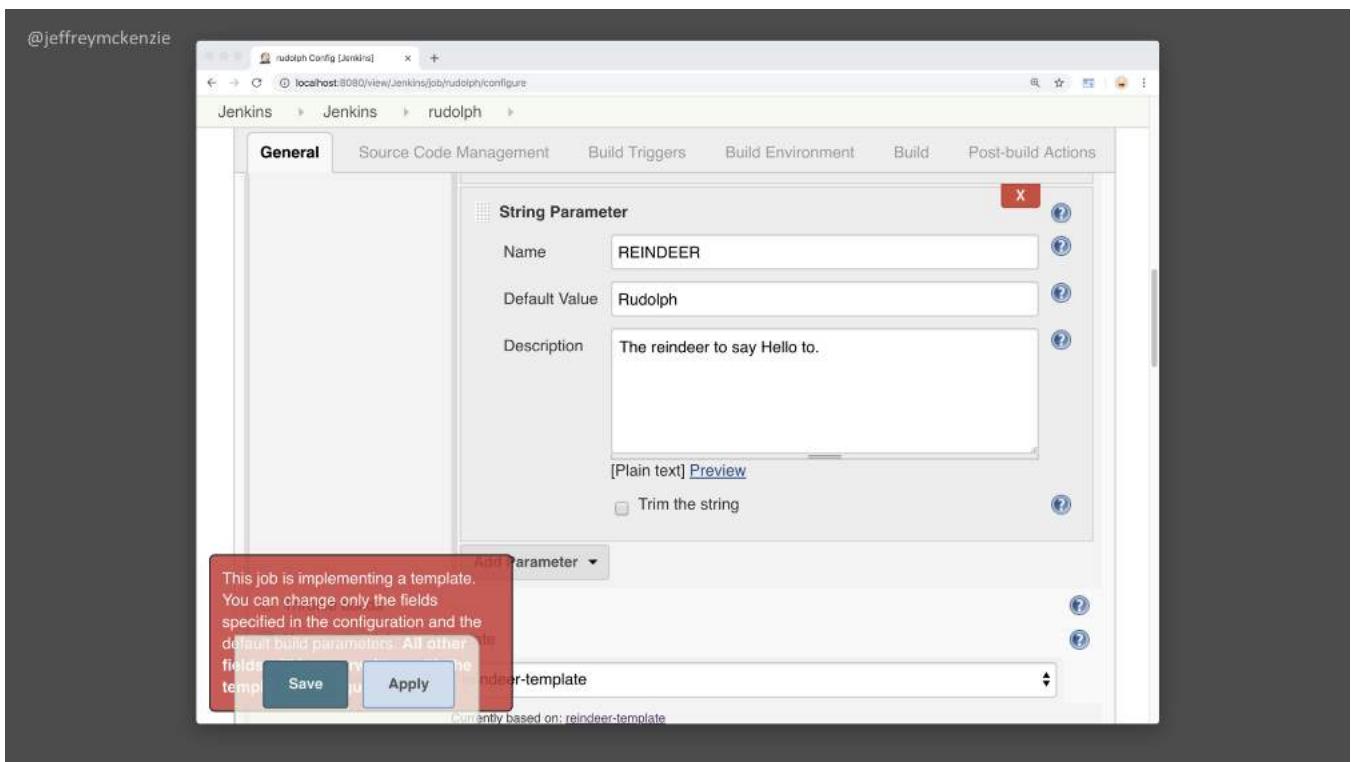
Our git settings...

@jeffreymckenzie



And our build step.

@jeffreymckenzie



So the only thing we have to do is
fill in the reindeer parameter
With Rudolph,
And run the project.

```
@jeffreymckenzie
rudolph # Jenkins [Console] + Jenkins job/rudolph/1/console
Jenkins   Jenkins   rudolph   #1
Commit message: "added greeting arg"
First time build. Skipping changelog.
[reindeer] $ /bin/sh -xe
/var/folders/35/khrsppd13w35brt4j20pgqr4cp_336/T/jenkins478907910
8618829720.sh
+ sh reindeer.sh Hello Rudolph
=====
=====
=
= Hello Rudolph!
=
=====
=====

Finished: SUCCESS

Page generated: Oct 2, 2018 12:44:38 PM EDT  REST API  Jenkins ver. 2.144
```

And our result is the same as before.

So now Santa can talk to as many reindeer as he wants,

And since they are all based on the same template,

The projects are much easier to change and maintain.

Those are the benefits, but there are limitations to this approach.

EZ Template Limitations

- ...cannot override all settings
- ...cannot be fully nested
- ...does not support pipeline jobs

First is not all settings can be overridden.

For example, if we wanted to modify

Our build step, our shell command, can only do so
In the template – local changes to that don't take effect

EZ Template Limitations

...cannot override all settings

...cannot be fully nested

...does not support pipeline jobs

Next, you can't really nest the templates effectively.

Would be nice to have a base template,

And have another template inherit from that.

But you can't, for example, add parameters to the child template,
Which would make it more useful.

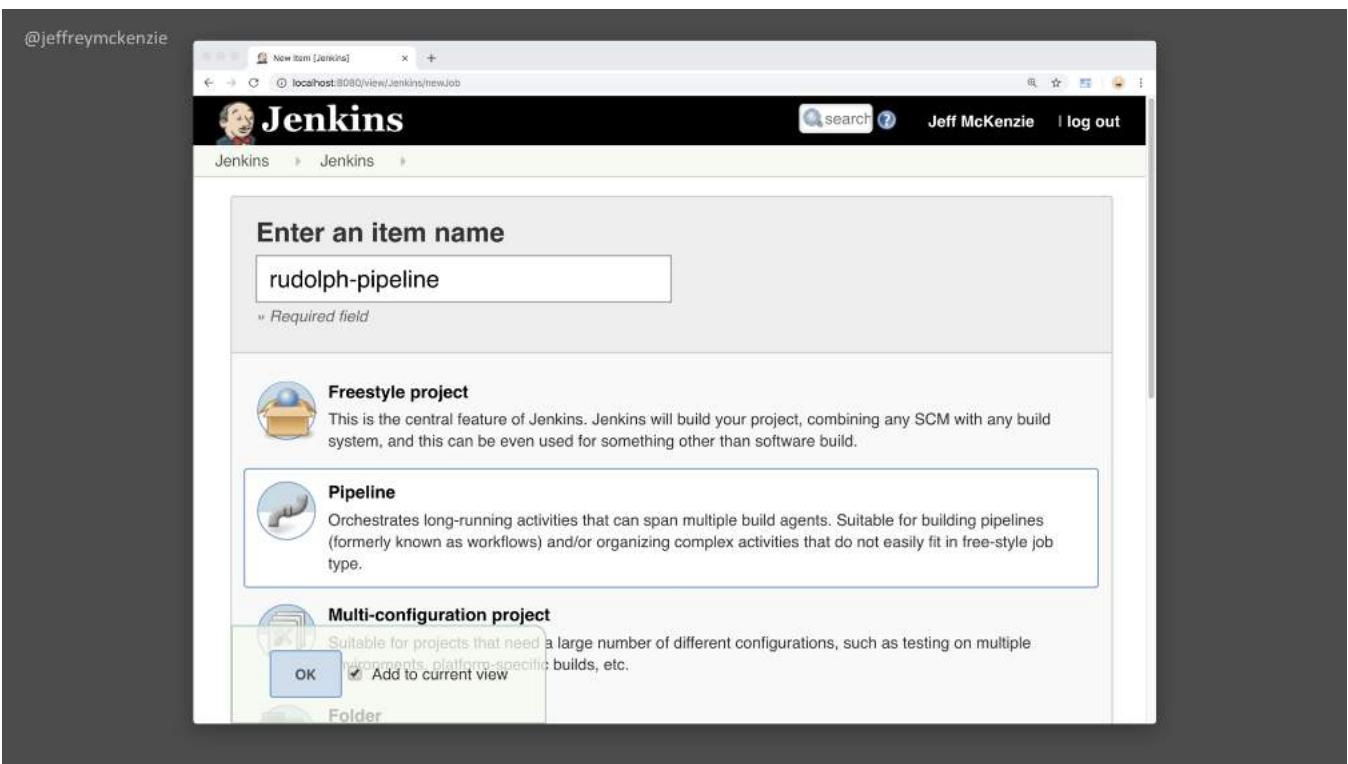
EZ Template Limitations

- ...cannot override all settings**
- ...cannot be fully nested**
- ...does not support pipeline jobs**

And lastly, templates only support freestyle jobs,
Not pipeline jobs.

Pipeline jobs is where the real power of Jenkins starts to come in.

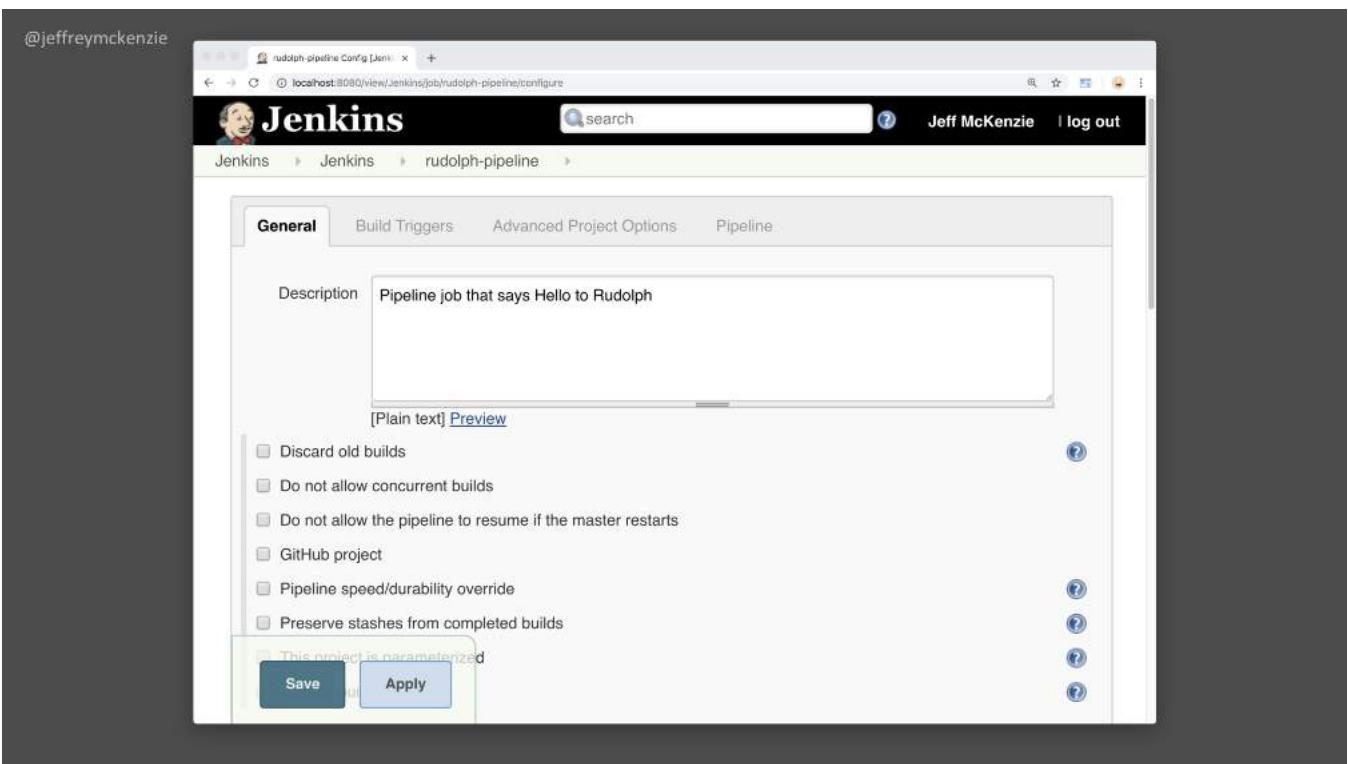
So templates are a great option where you have a lot of freestyle jobs
That exhibit a similar pattern, and where parameters would be helpful.



So let's turn to our next option for scaling,
The pipeline job.

Let's create one called Rudolph pipeline
that does the same thing
Our freestyle job does, to show the differences.

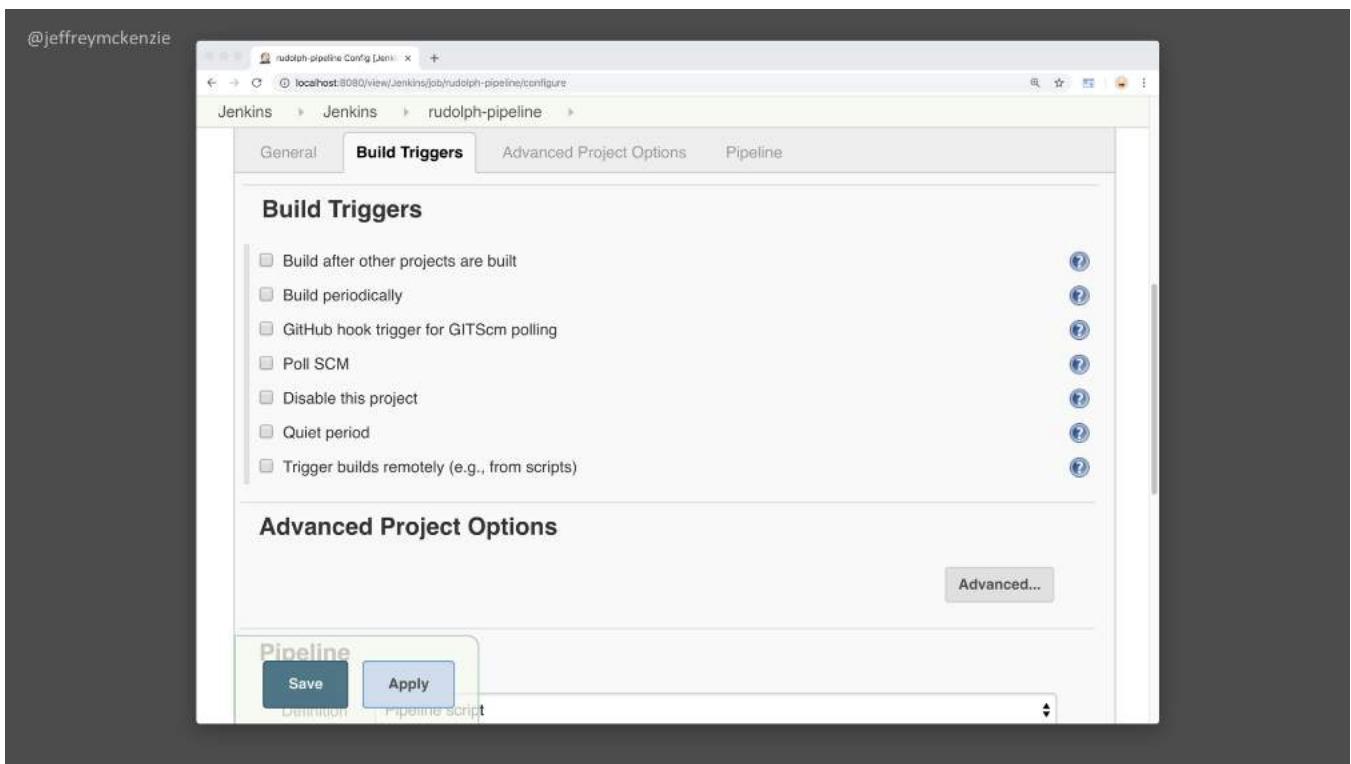
-- comment on pipeline description



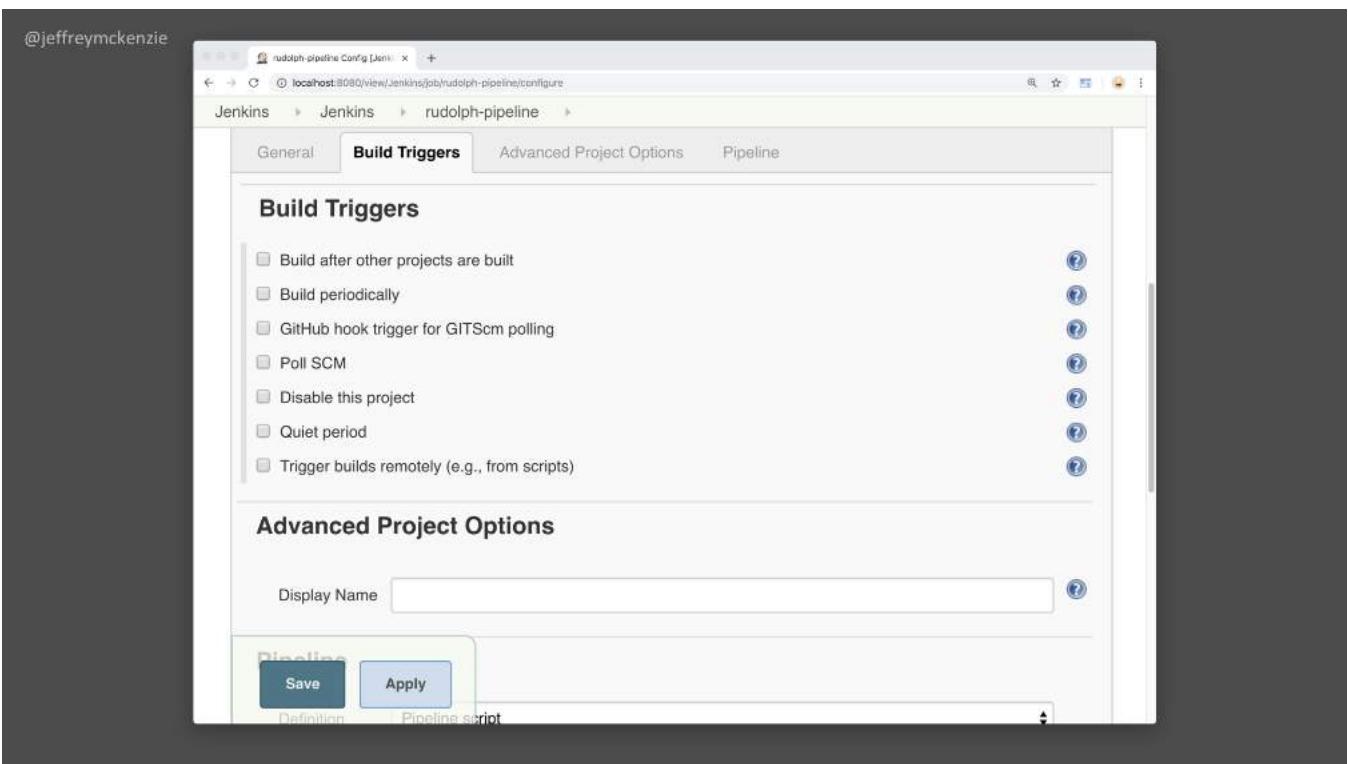
We will add our description,
And notice that this has less configuration options
Than a freestyle build.

We have the General section...

@jeffreymckenzie

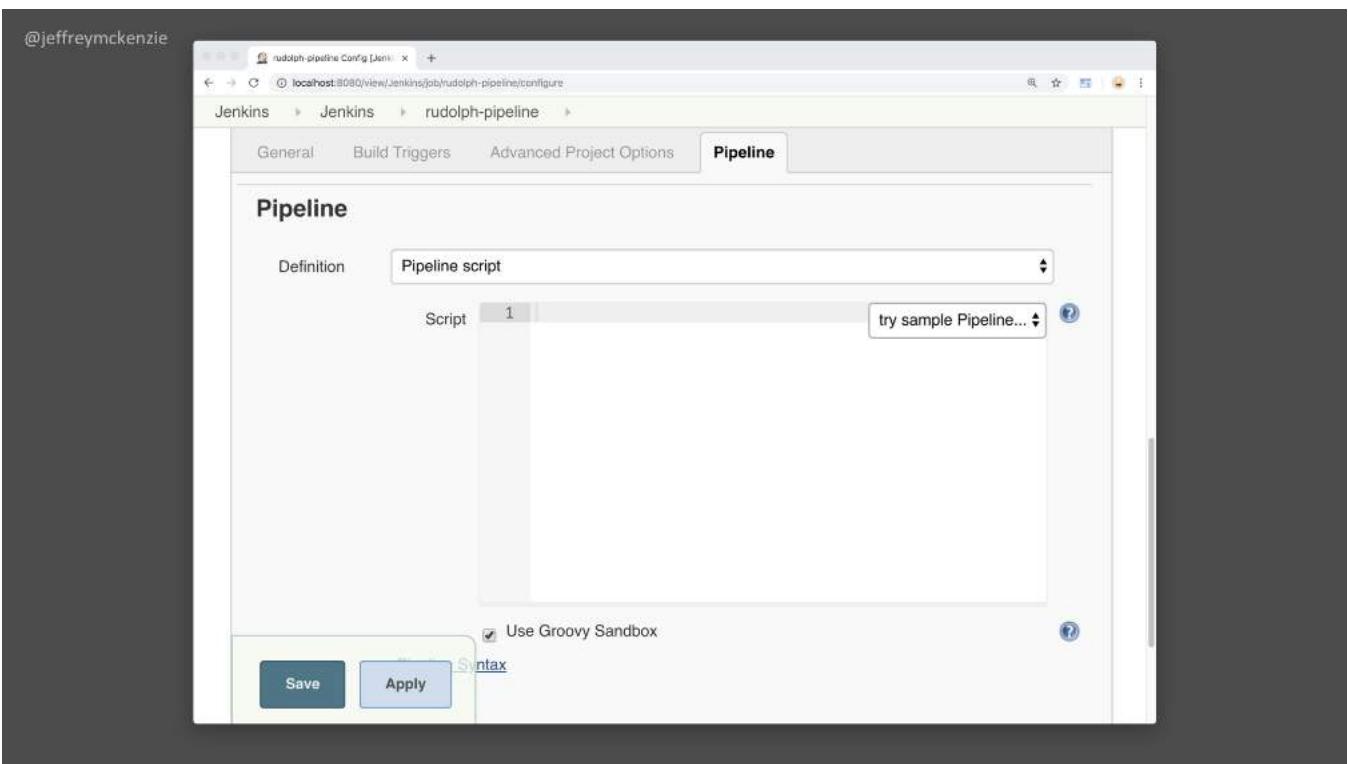


The build triggers section,
And the advanced project options.



Btw I really like that the advanced project options
Is display name – really?

But the main part of the pipeline job is the script



The script allows you to do the same thing
As a freestyle job, except you can do it
Programmatically, in a concise way.

Let's take a close-up look at that script

@jeffreymckenzie

```
pipeline {
    agent any
    options {
        buildDiscarder(logRotator(numToKeepStr: '3'))
    }
    parameters {
        string{
            name: 'GREETING',
            defaultValue: 'Hello',
            description: 'The way to say hello.'
        }
        string{
            name: 'REINDEER',
            defaultValue: 'Rudolph',
            description: 'The reindeer to say hello to.'
        }
    }
    stages {
        stage('Say Hi to Rudolph') {
            steps {
                checkout scm: [
                    $class: 'GitSCM',
                    branches: [
                        [name: '*/master']
                    ],
                    userRemoteConfigs: [
                        [url: 'file:///Users/jmckenzie/projects/santa']
                    ]
                ]
                sh 'sh reindeer.sh "${GREETING}" "${REINDEER}"'
            }
        }
    }
}
```

Here's the whole thing – As you can see, not a lot of code here
Using a DSL – what is a DSL?

There are two different syntax formats for pipeline scripts

- Scripting syntax
- Declarative syntax (newer)

With scripting syntax you tell Jenkins how to perform a task
With declarative, you tell Jenkins what you want to do.

Using declarative here, let's break this down

@jeffreymckenzie

```
pipeline {  
    agent any  
    options {  
        buildDiscarder(logRotator(numToKeepStr:'3'))  
    }  
}
```

On the Jenkins site, there's a reference area
for all of these commands and syntax,
So you can look all this stuff up

A pipeline script has to start with the pipeline block.
Then you specify the agent, or where this is going to run

For our purposes this can run anywhere...

We can set the build history to keep the last 3 builds,
just like we did in the freestyle job

@jeffreymckenzie

```
pipeline {  
    agent any  
    options {  
        buildDiscarder(logRotator(numToKeepStr:'3'))  
    }  
    parameters {  
        string( name: 'GREETING',  
               defaultValue: 'Hello',  
               description: 'The way to say hello. ')  
    }  
}
```

Then we can add the greeting parameter,
Again, same settings and values

@jeffreymckenzie

```
pipeline {  
    agent any  
    options { ... }  
    parameters {  
        ...  
        string( name: 'REINDEER',  
               defaultValue: 'Rudolph',  
               description: 'The reindeer to say hello to.')  
    }  
}
```

Add our reindeer parameter...

@jeffreymckenzie

```
pipeline {  
    agent any  
    options { ... }  
    parameters { ... }  
    stages {  
        stage('Say Hi to Rudolph') {  
            steps {  
            }  
        }  
    }  
}
```

Then we add the stages block,
Which is just a way to group commands into sections.

We have only one stage, and we can give it a name...
And within a stage, we have steps.

@jeffreymckenzie

```
pipeline {  
    ...  
    stage('Say Hi to Rudolph') {  
        steps {  
            checkout scm: [$class: 'GitSCM',  
                branches: [[name: '*/master']],  
                userRemoteConfigs: [[url:  
                    'file:///Users/jmckenzie/projects/santa']]  
        }  
    }  
}
```

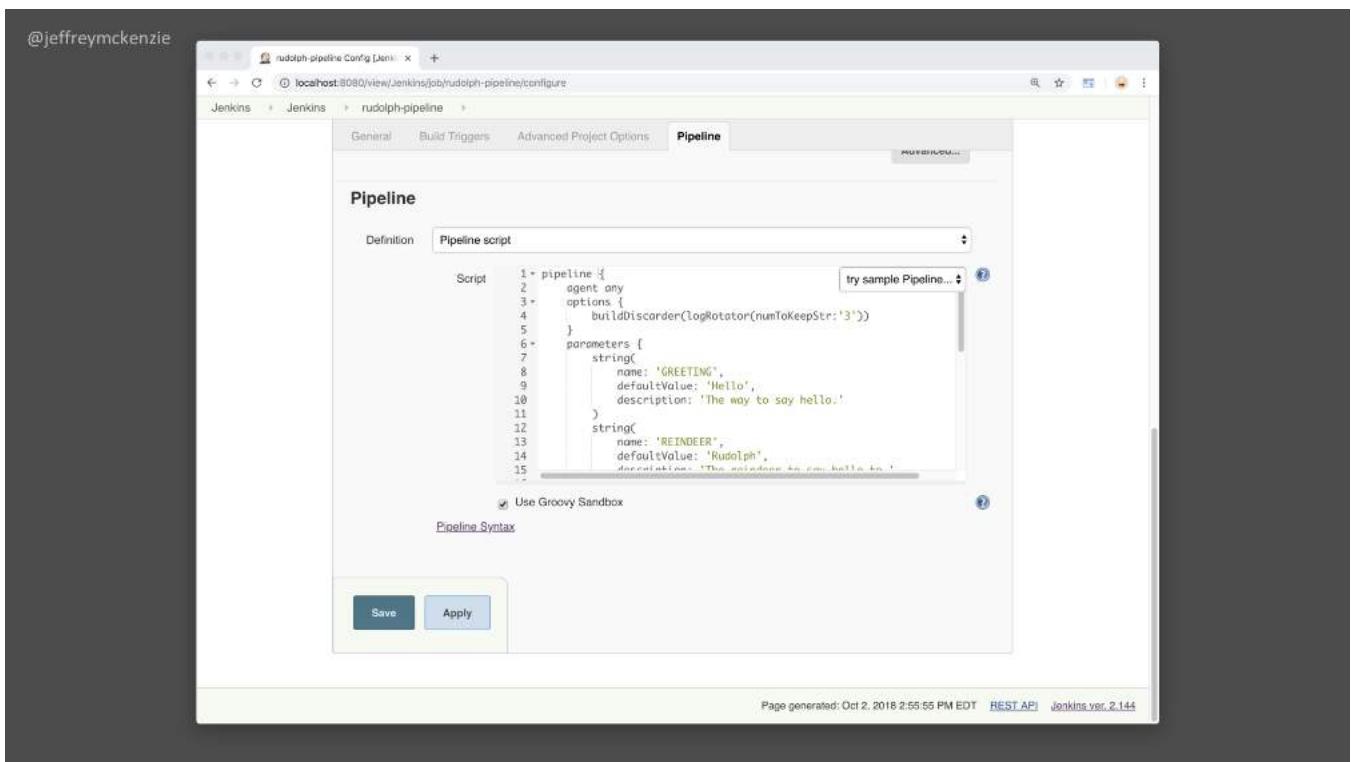
Our first step will be do to the checkout
From the git repository

@jeffreymckenzie

```
pipeline {  
    ...  
    stage('Say Hi to Rudolph') {  
        steps {  
            checkout scm:[ ... ]  
            sh 'sh reindeer.sh  
                "${GREETING}" "${REINDEER}"'  
        }  
    }  
}
```

And the second step is to execute the shell script
Using the SH command and our parameters.

@jeffreymckenzie



So here's what it looks like back in the pipeline project...

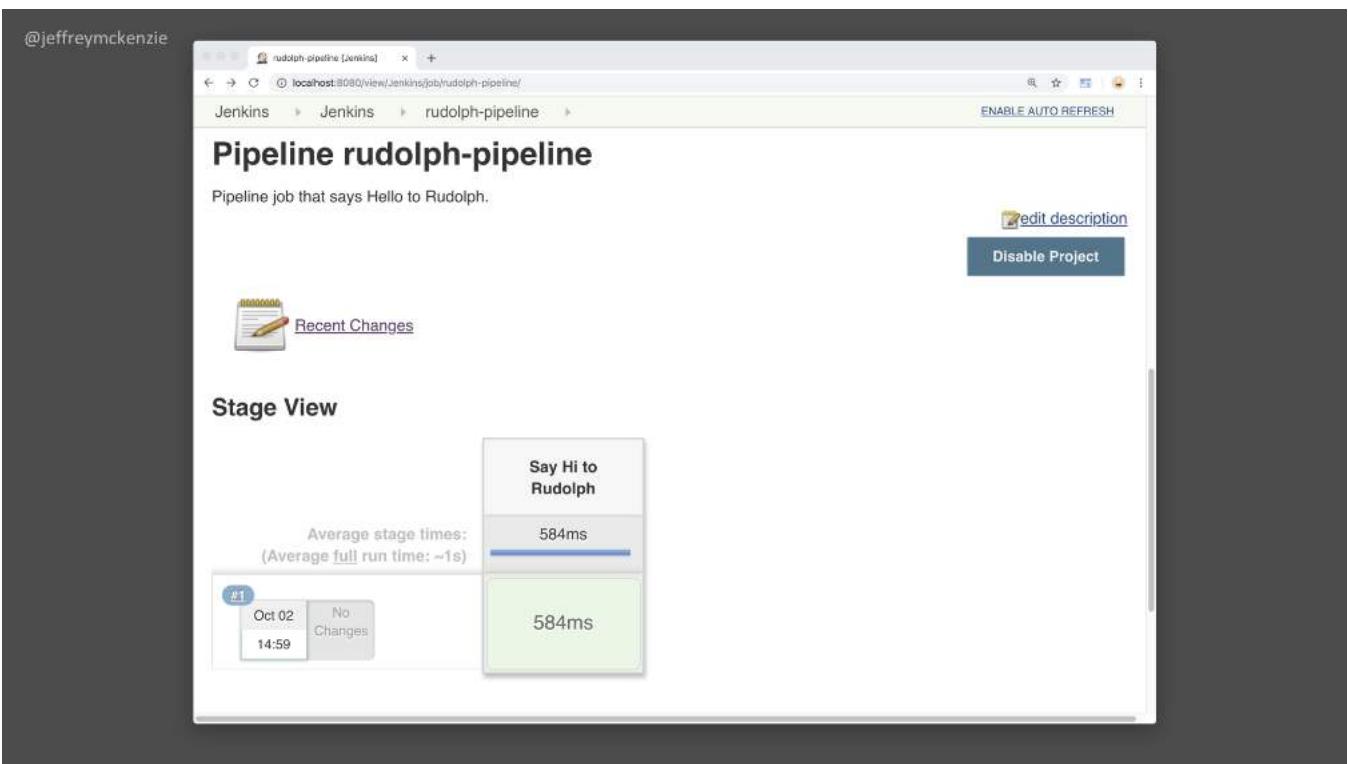
And let's run that and see what we get

@jeffreymckenzie

The screenshot shows the Jenkins interface for the pipeline job 'rudolph-pipeline'. The top navigation bar includes links for 'Jenkins', 'Jenkins', 'rudolph-pipeline', 'Search', 'Jeff McKenzie', and 'log out'. A 'Pipeline' icon is also present. On the left, a sidebar lists options: 'Back to Dashboard', 'Status', 'Changes', 'Build Now', 'Delete Pipeline', 'Configure', 'Full Stage View', 'Rename', and 'Pipeline Syntax'. Below this is a search bar with 'find' and a 'trend' dropdown set to 'recent'. A 'Build History' section shows a single build: '#1 Oct 2, 2018 2:59 PM'. It includes links for 'RSS for all' and 'RSS for failures'. To the right, the main content area displays the pipeline stages. The first stage, 'Say Hi to Rudolph', is shown with a duration of '584ms'. Below it, a 'Stage View' section indicates an average stage time of '584ms' and an average full run time of '~1s'. The stage view shows a single step with a timestamp of 'Oct 02 14:59' and 'No Changes'. At the bottom, there are 'Permalinks' and footer text: 'Page generated: Oct 2, 2018 2:59:08 PM EDT REST API Jenkins ver. 2.144'.

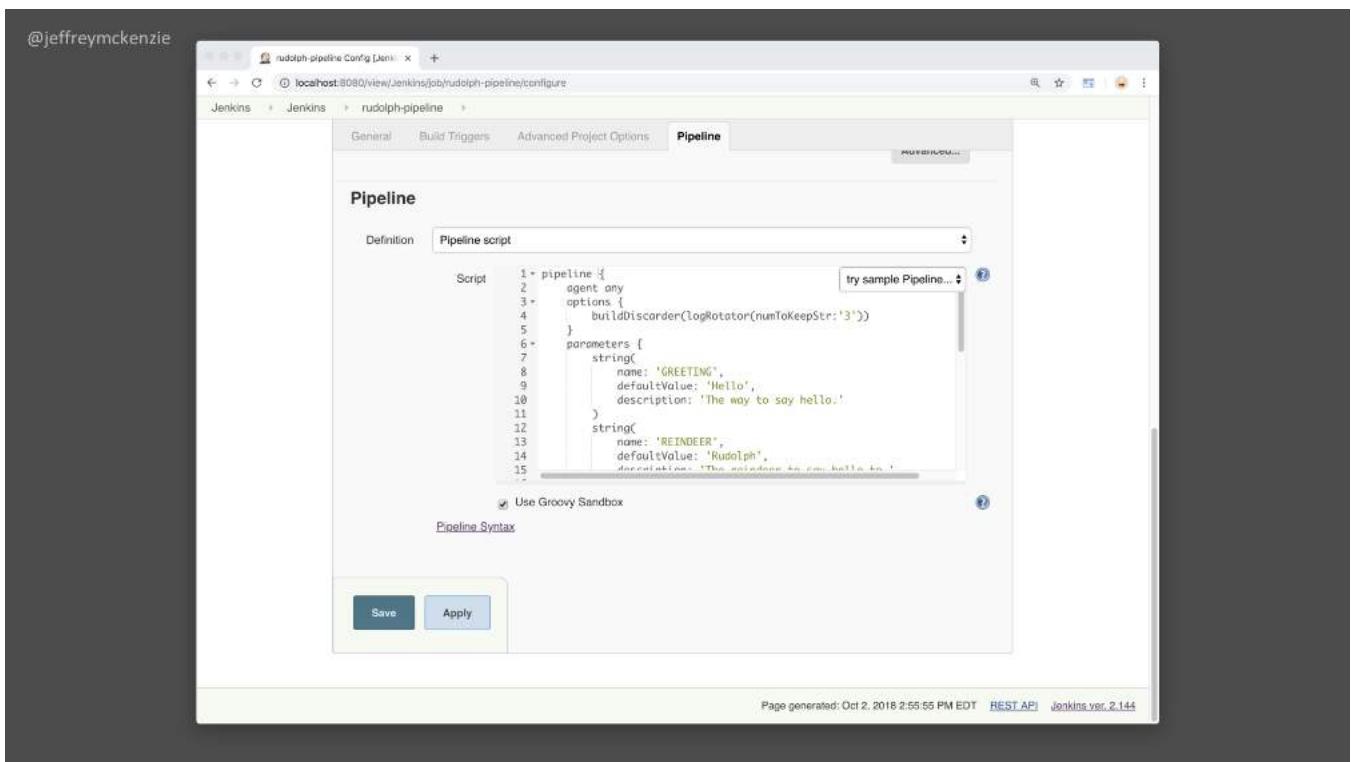
We have a successful build, and if you'll notice
There's a stage view here as well....

@jeffreymckenzie



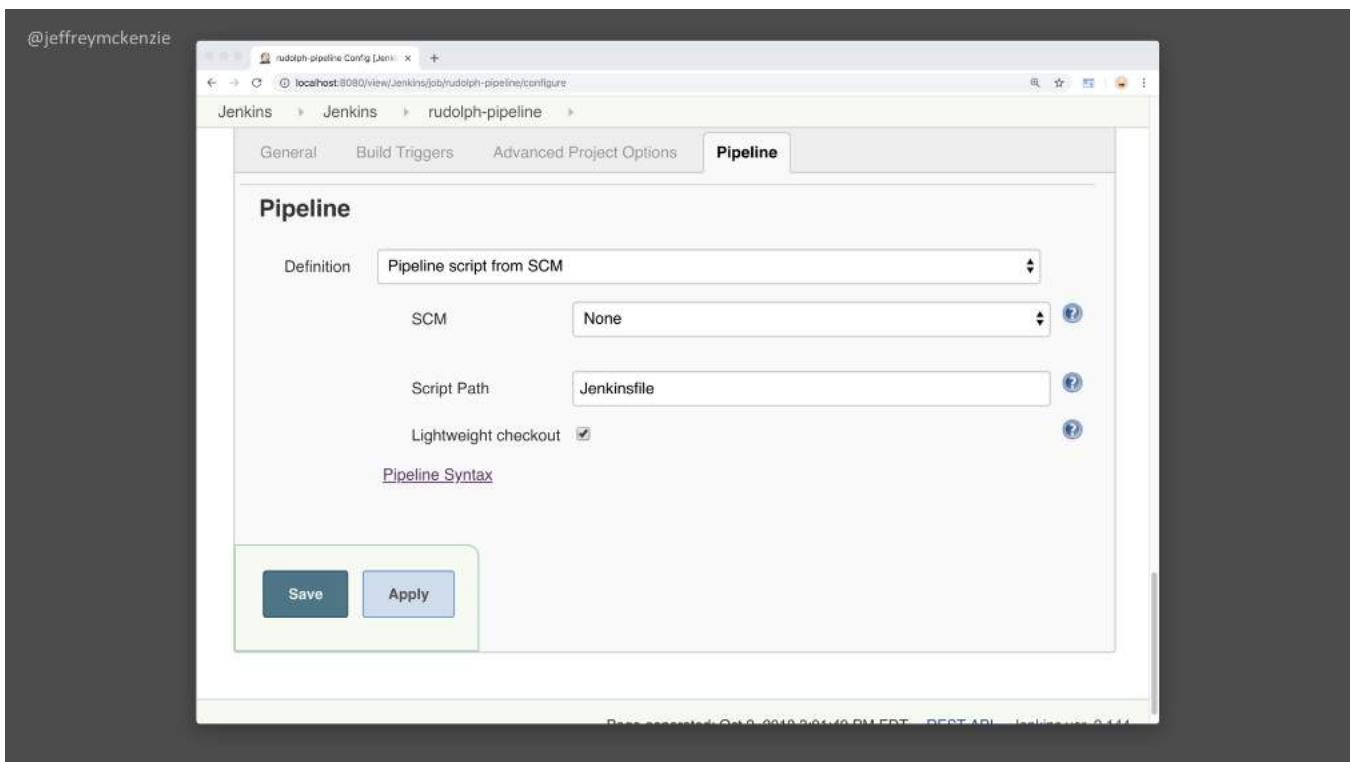
Which shows the name of the stage
And how long it took to run.

@jeffreymckenzie



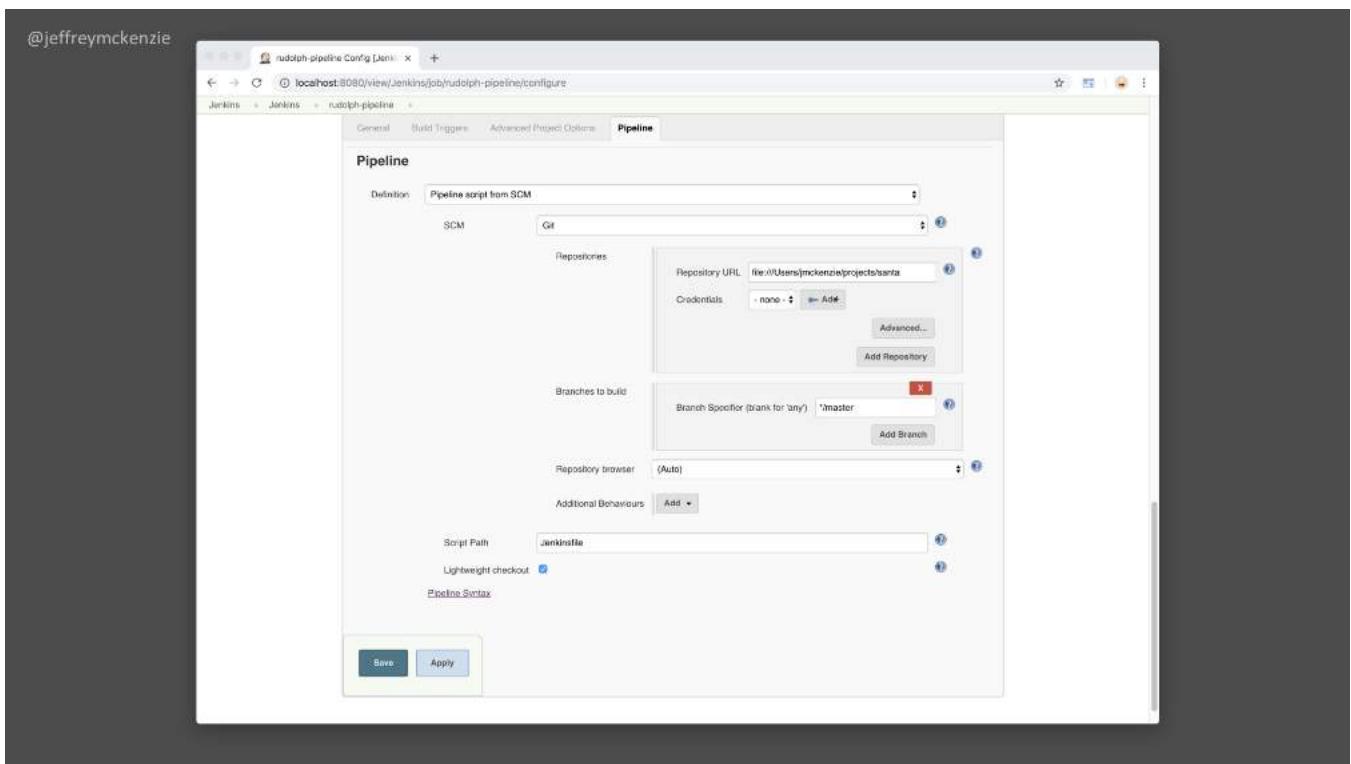
If we go back to our pipeline,
There's another option here for definition...

@jeffreymckenzie



Which is “pipeline script from SCM”
Or source control management,

@jeffreymckenzie



We can choose a git repo where this is coming from,
Which is great because it allows us
To version control that pipeline script
Just like we would with any other code.

@jeffreymckenzie

```
pipeline {
    agent any
    options {
        buildDiscarder(logRotator(numToKeepStr:3))
    }
    parameters {
        string{
            name: 'GREETING',
            defaultValue: 'Hello',
            description: 'The way to say hello.'
        }
        string{
            name: 'REINDEER',
            defaultValue: 'Rudolph',
            description: 'The reindeer to say hello to.'
        }
    }
    stages {
        stage('Say Hi to Rudolph') {
            steps {
                checkout scm: [
                    $class: 'GitSCM',
                    branches: [
                        [name: '*/*master']
                    ],
                    userRemoteConfigs: [
                        [url: 'file:///Users/jmckenzie/projects/santa']
                    ]
                ]
                sh 'sh reindeer.sh "${GREETING}" "${REINDEER}"'
            }
        }
    }
}
```

So that's great and everything,
But Santa still has a problem –

@jeffreymckenzie



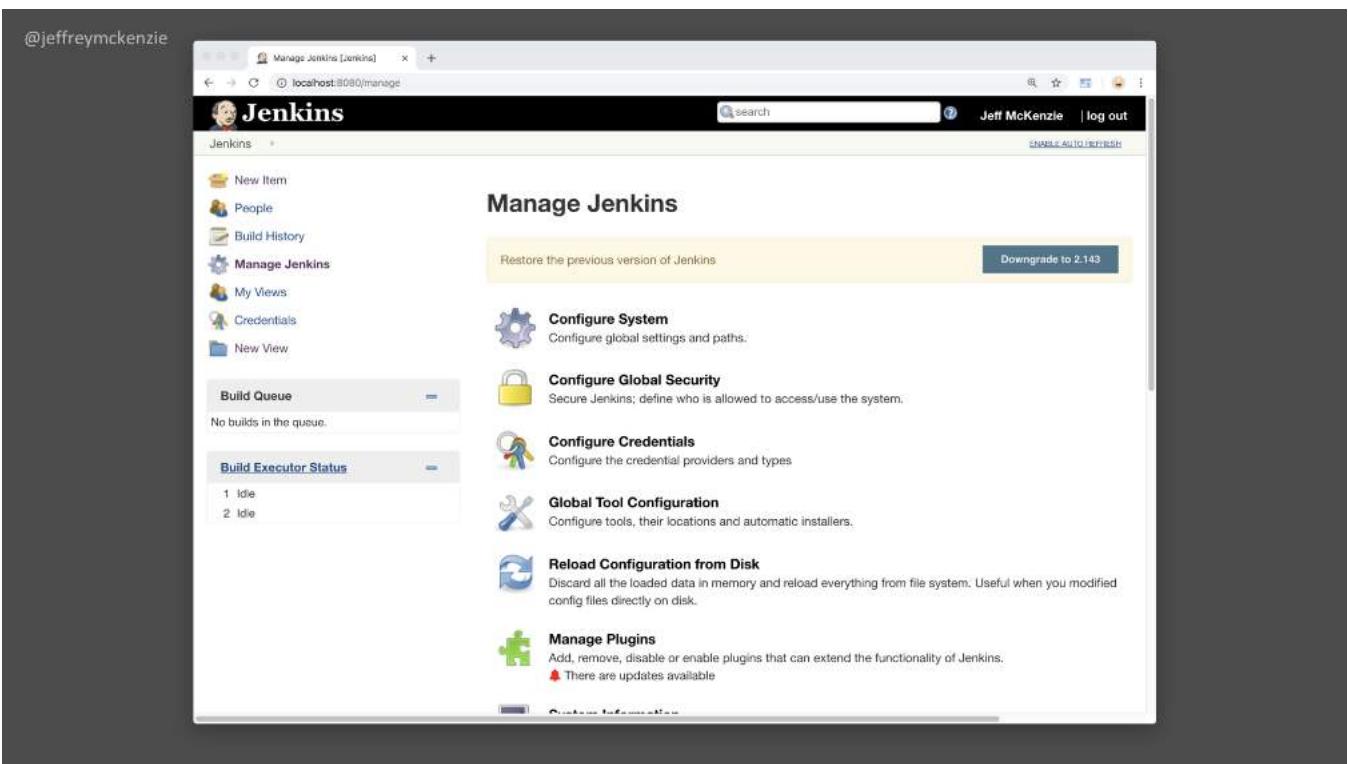
He has to copy this script
to any pipeline project that needs it.

So let's fix that.

[https://commons.wikimedia.org/wiki/File:Mr_Santa_Claus_\(HS85-10-30308\).jpg](https://commons.wikimedia.org/wiki/File:Mr_Santa_Claus_(HS85-10-30308).jpg)

British Library [Public domain or Public domain], via Wikimedia Commons

@jeffreymckenzie

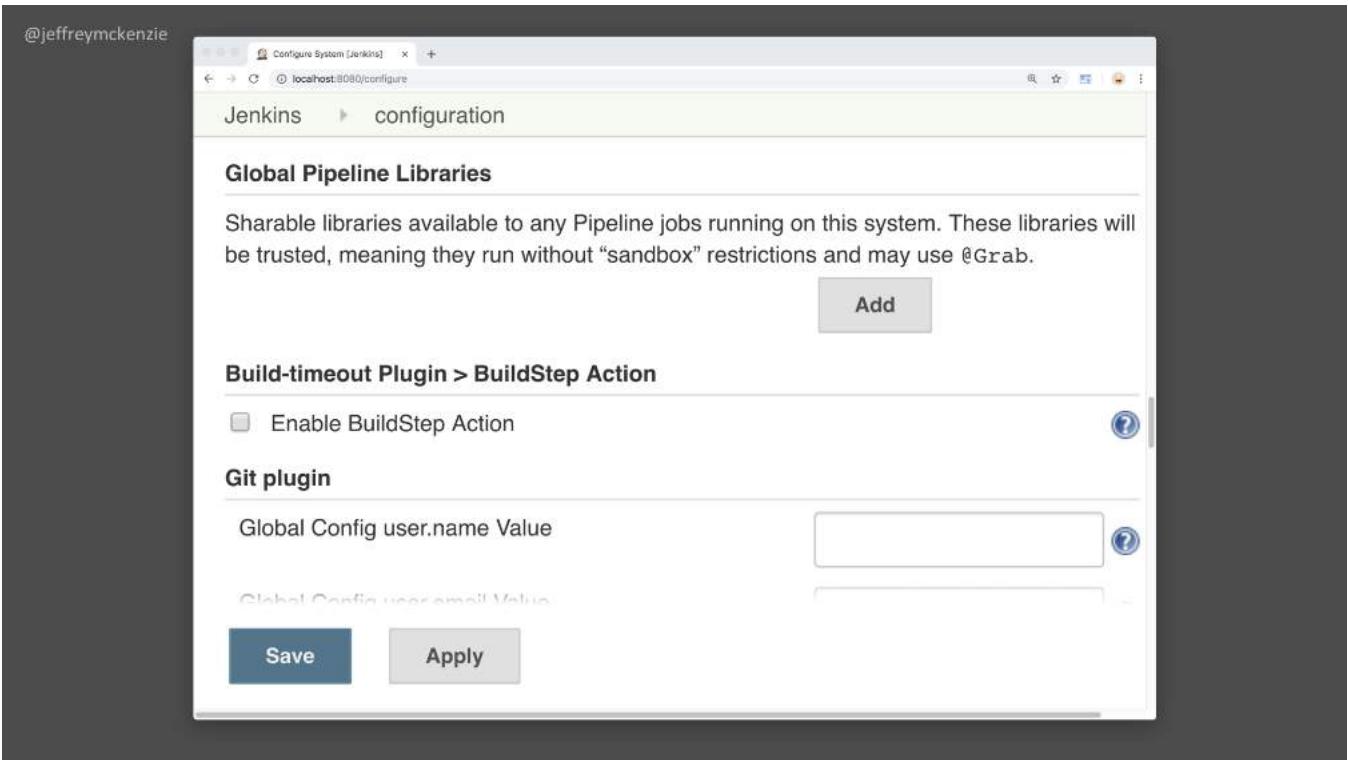


If we go to the manage Jenkins page,
Into Configure System...

@jeffreymckenzie

The screenshot shows the Jenkins 'Configure System' page. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', 'New View', 'Build Queue' (which says 'No builds in the queue.'), and 'Build Executor Status' (which shows '1 Idle' and '2 Idle'). The main content area is titled 'Global Pipeline Libraries'. It has a 'Home directory' field set to '/Users/jmckenzie/.jenkins'. Below it are sections for 'Labels' (set to '2'), 'Usage' (set to 'Use this node as much as possible'), 'Quiet period' (set to '5'), 'SCM checkout retry count' (set to '0'), 'Default view' (set to 'all'), and 'Restrict project naming'. There's also a 'Global properties' section with checkboxes for 'Environment variables' and 'Tool Locations'. At the bottom, there's a 'Pipeline Speed/Durability Settings' section with a dropdown set to 'None: use pipeline default (MAX_SURVIVABILITY)'. At the very bottom are 'Save' and 'Apply' buttons.

And we scroll down a bit,
We get to a section called Global Pipeline Libraries



Which says...

Shareable libraries available to any Pipeline jobs

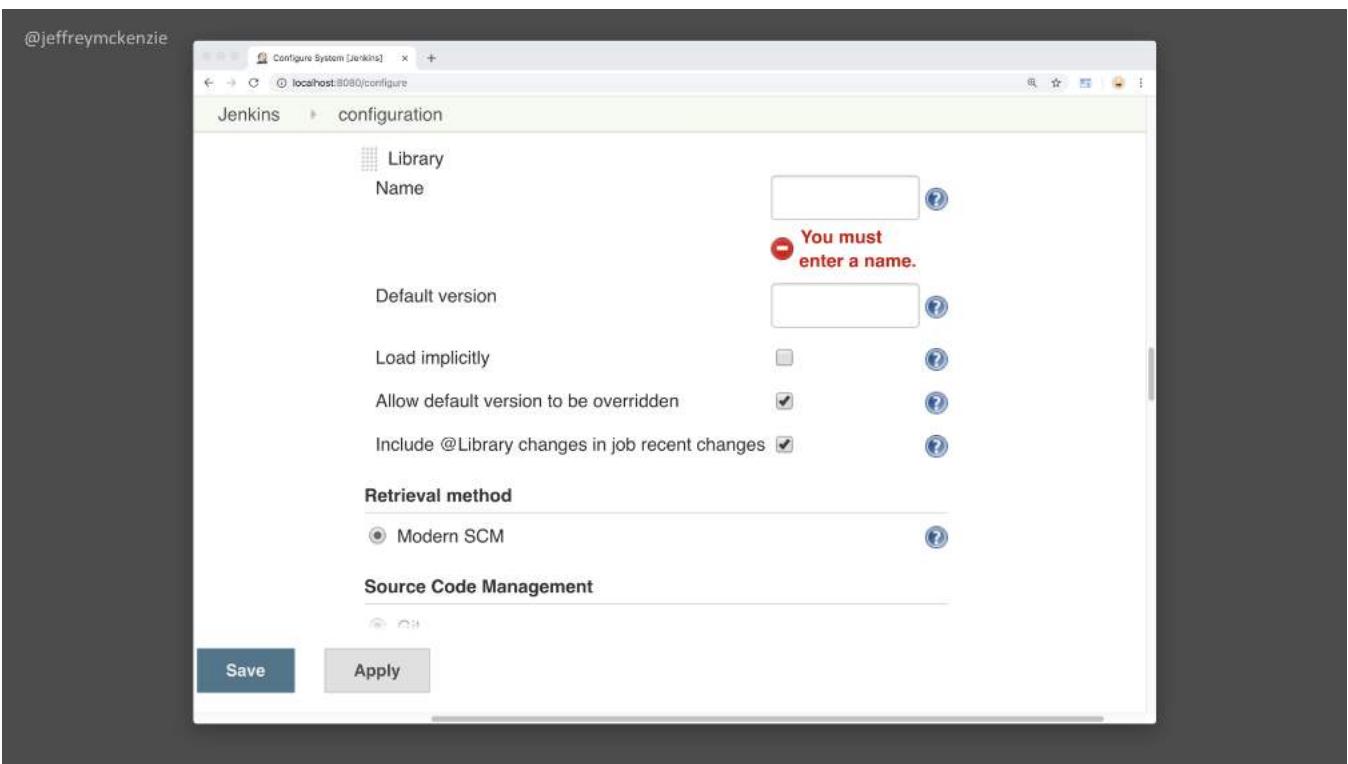
Running on this system.

That means we can create shared code

(written in groovy)

Accessible to any pipeline script.

To do that, we click add...



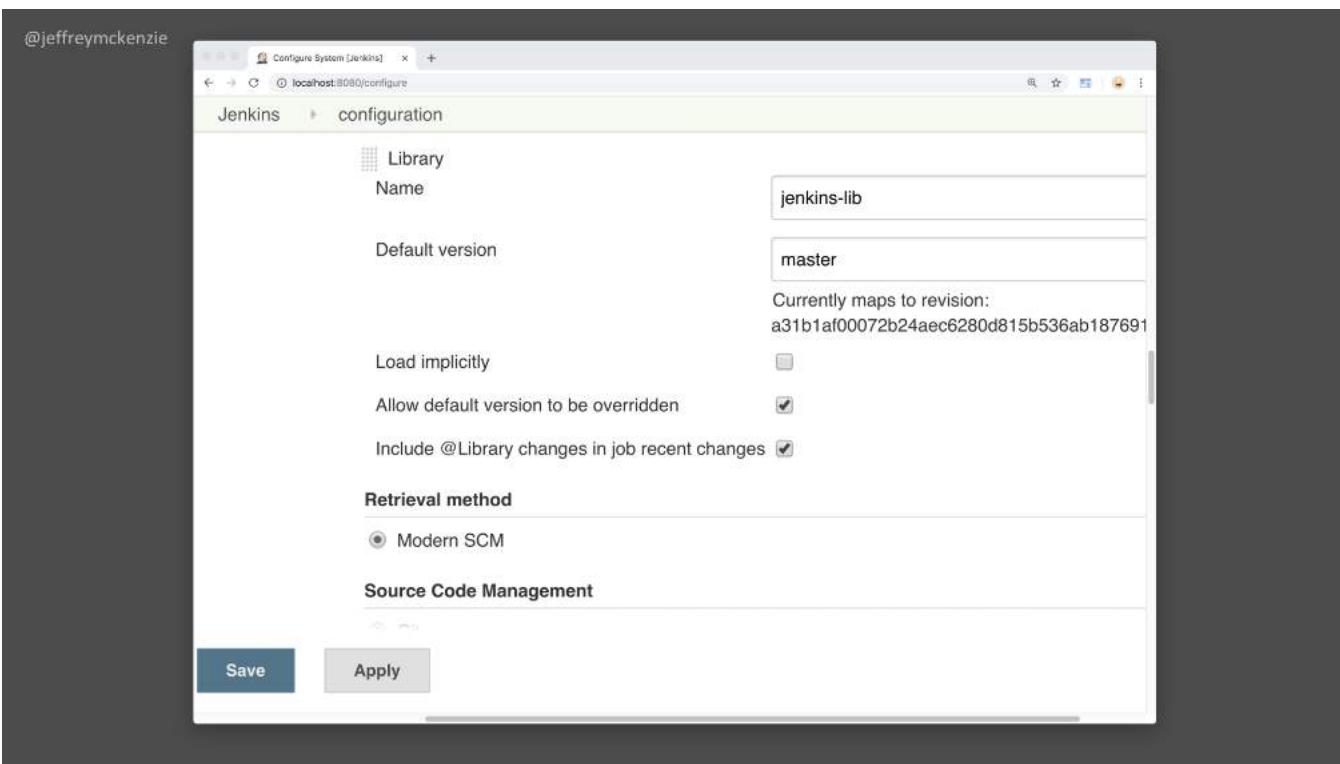
Then you'll add information about your library
-- a name: we'll call it Jenkins-lib

@jeffreymckenzie

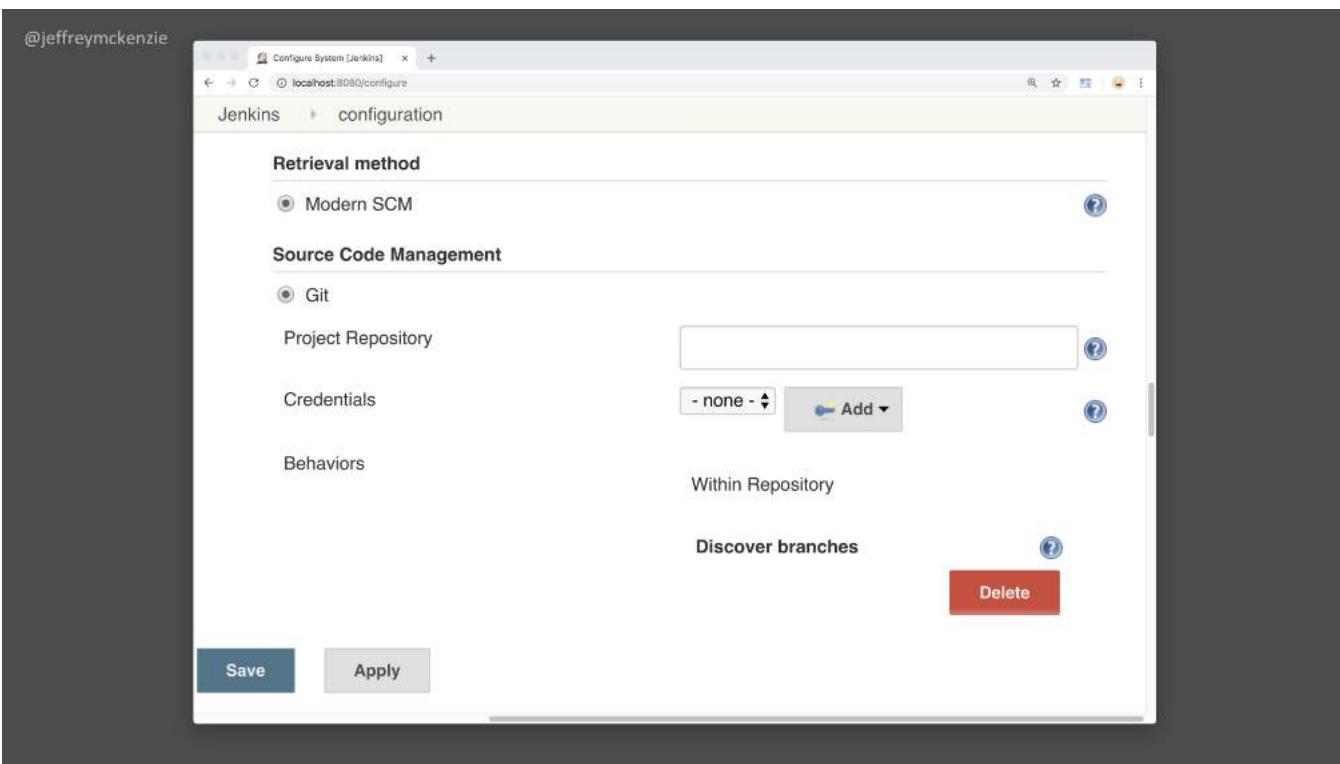
The screenshot shows the Jenkins configuration interface for a library named "jenkins-lib". The configuration includes:

- Name:** jenkins-lib
- Default version:** (empty field)
- Load implicitly:** (unchecked checkbox)
- Allow default version to be overridden:** (checked checkbox)
- Include @Library changes in job recent changes:** (checked checkbox)
- Retrieval method:** Modern SCM (radio button selected)
- Source Code Management:** Git (radio button selected)
- Project Repository:** (empty field)

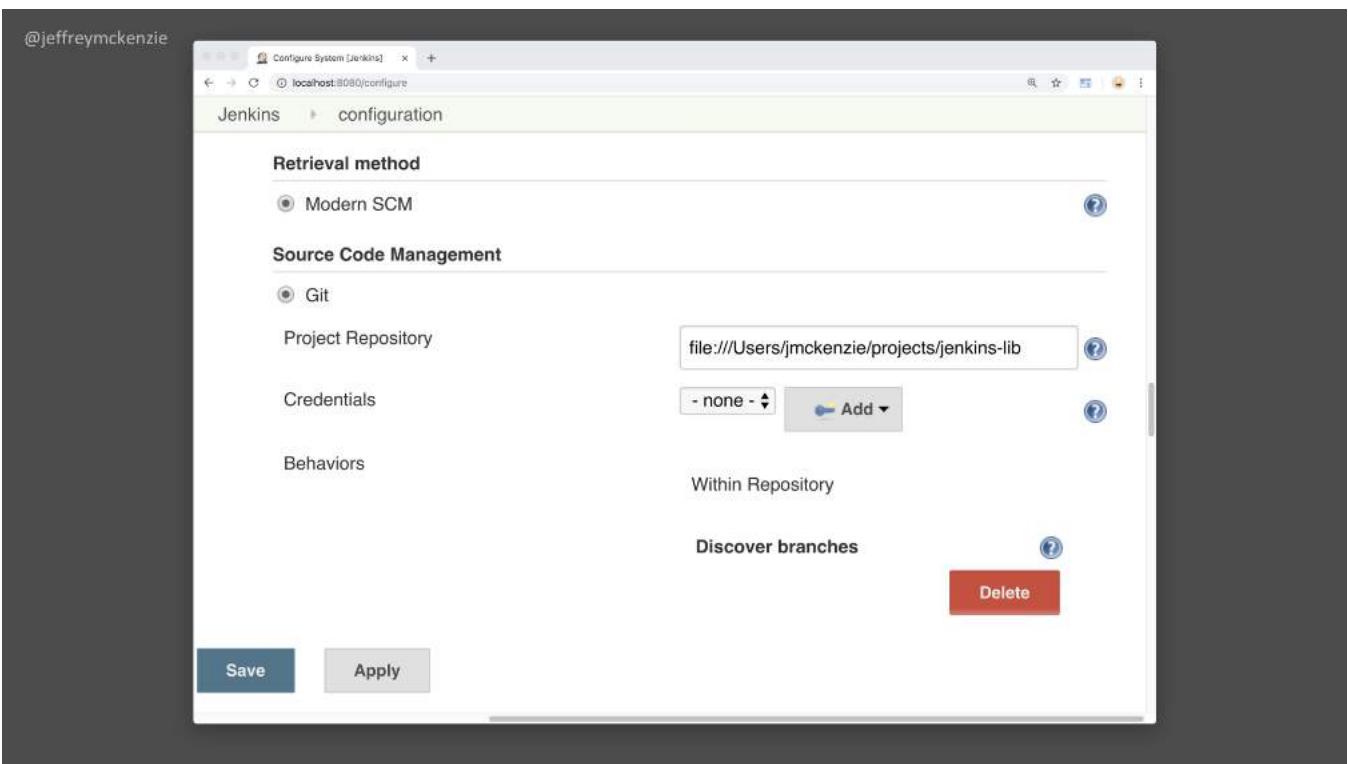
At the bottom are "Save" and "Apply" buttons.



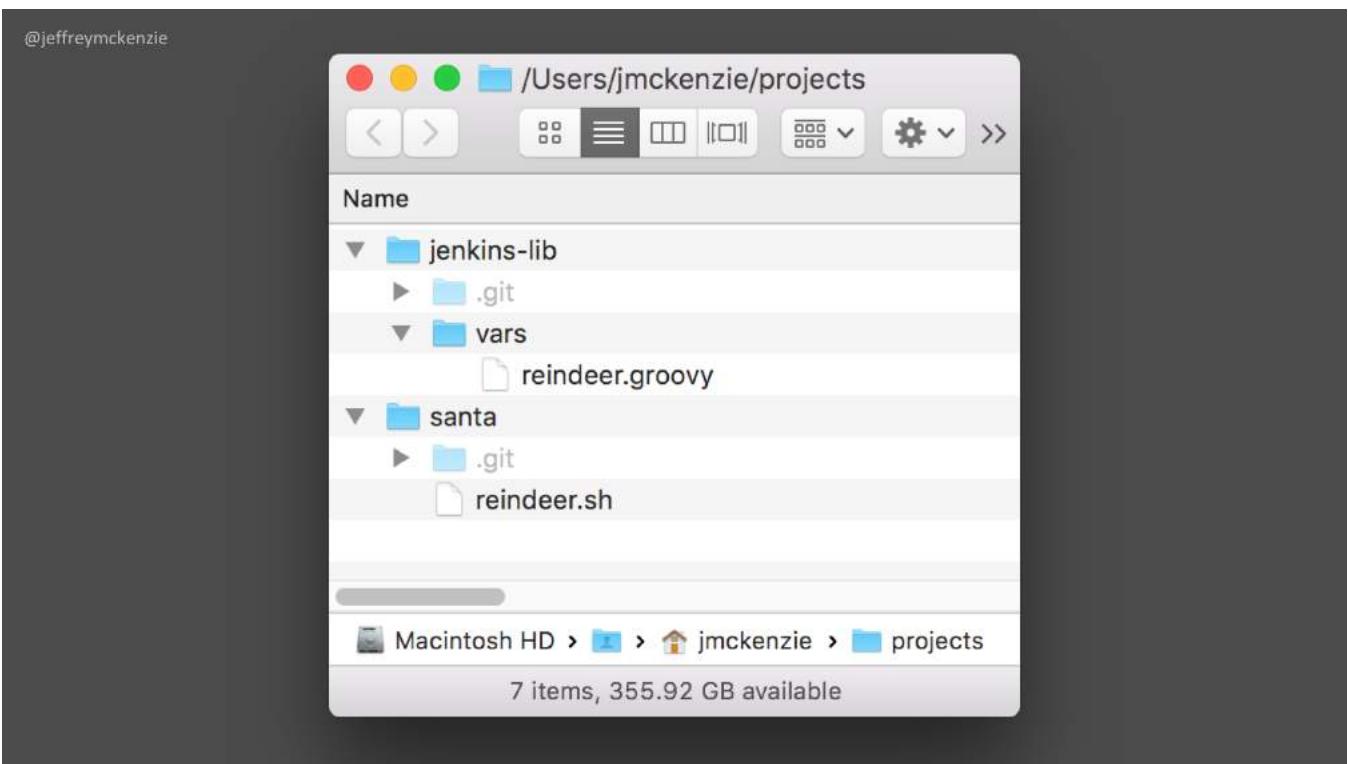
Default version refers to version control,
So we will just list the master branch



Then we add the git repo
we want to pull the shared code from



Again we'll point that to a separate git repo locally
And click save



So these are our 2 git repos –
Santa, which contains our shell script,

And Jenkins-lib, which contains
Our global pipeline library

Jenkins expects to see a directory
Called VARS, with groovy scripts
underneath that

-- let's look at the reindeer groovy script
in the library

@jeffreymckenzie

```
- reindeer.groovy

#!/usr/bin/env groovy
def call(Map<String, Object> options) {
    try {
        node {
            String greeting = options."greeting".toString()
            String reindeer = options."reindeer".toString()
            stage("Say Hi to ${reindeer}") {
                checkout scm: [
                    $class: 'GitSCM',
                    branches: [[name: '*/master']],
                    userRemoteConfigs: [[url:
                        'file:///Users/jmckenzie/projects/santa']]]
                sh "sh reindeer.sh ${greeting} ${reindeer}"
            }
        }
        catch (Throwable err) {
            throw err
        }
    }
}
```

so this is a rewrite of our pipeline script in groovy
Groovy is a scripting language
Similar to ruby,
that runs on the JVM

This is even less code than our pipeline script.
Let's look at it in more detail.

@jeffreymckenzie

- reindeer.groovy

```
#!/usr/bin/env groovy
def call(Map<String, Object> options) {
    try {
        node {

        }
    }
    catch (Throwable err) {
        throw err
    }
}
```

You create a method using the DEF keyword,
And then name your method “call” --
You refer to the method in Jenkins by the file name
Rather than the actual method name,
Which we will see in a little bit.

We’re going to pass in a Map object called “options”
Which is simply a key-value pair collection.
We can use try and catch,
And the node block in where we place
What will run in the pipeline

@jeffreymckenzie

```
- reindeer.groovy

#!/usr/bin/env groovy
def call(Map<String, Object> options) {
    try {
        node {
            String greeting =
                options."greeting".toString()
            String reindeer =
                options."reindeer".toString()
        }
    }
    catch (Throwable err) { ... }
}
```

Then we declare two variables –
Greeting and reindeer, that hold our parameters.

@jeffreymckenzie

```
- reindeer.groovy

def call(Map<String, Object> options) {
    ...
    String greeting = ...
    String reindeer = ...
    stage("Say Hi to ${reindeer}") {
        ...
    }
}
catch (Throwable err) { ... }
```

Then we add the stage block –
And here we can actually incorporate
The variable name into the Jenkins output

@jeffreymckenzie

```
- reindeer.groovy

def call(Map<String, Object> options) {
    ...
    stage("Say Hi to ${reindeer}") {
        checkout scm: [
            $class: 'GitSCM',
            branches: [[name: '*/master']],
            userRemoteConfigs: [[url:'file:///Users/jmckenzie/projects/santa']]]
        }
        ...
    }
}
```

Then the git checkout from our santa repo,
to pull our reindeer shell script

@jeffreymckenzie

```
- reindeer.groovy

def call(Map<String, Object> options) {
    ...
    stage("Say Hi to ${reindeer}") {
        checkout scm: [ ... ]
        sh "sh reindeer.sh ${greeting}
            ${reindeer}"
    }
    ...
}
```

And finally our call to the shell script,
Where we can pass the variables in.

Let's create a new pipeline job and see how this works.

@jeffreymckenzie

The screenshot shows the Jenkins web interface for creating a new item. The title bar says "New Item [Jenkins]". The main header has the Jenkins logo and the text "Jenkins" and "Jeff McKenzie | log out". Below the header, there's a breadcrumb navigation: "Jenkins > All >". A search bar and a help icon are also present.

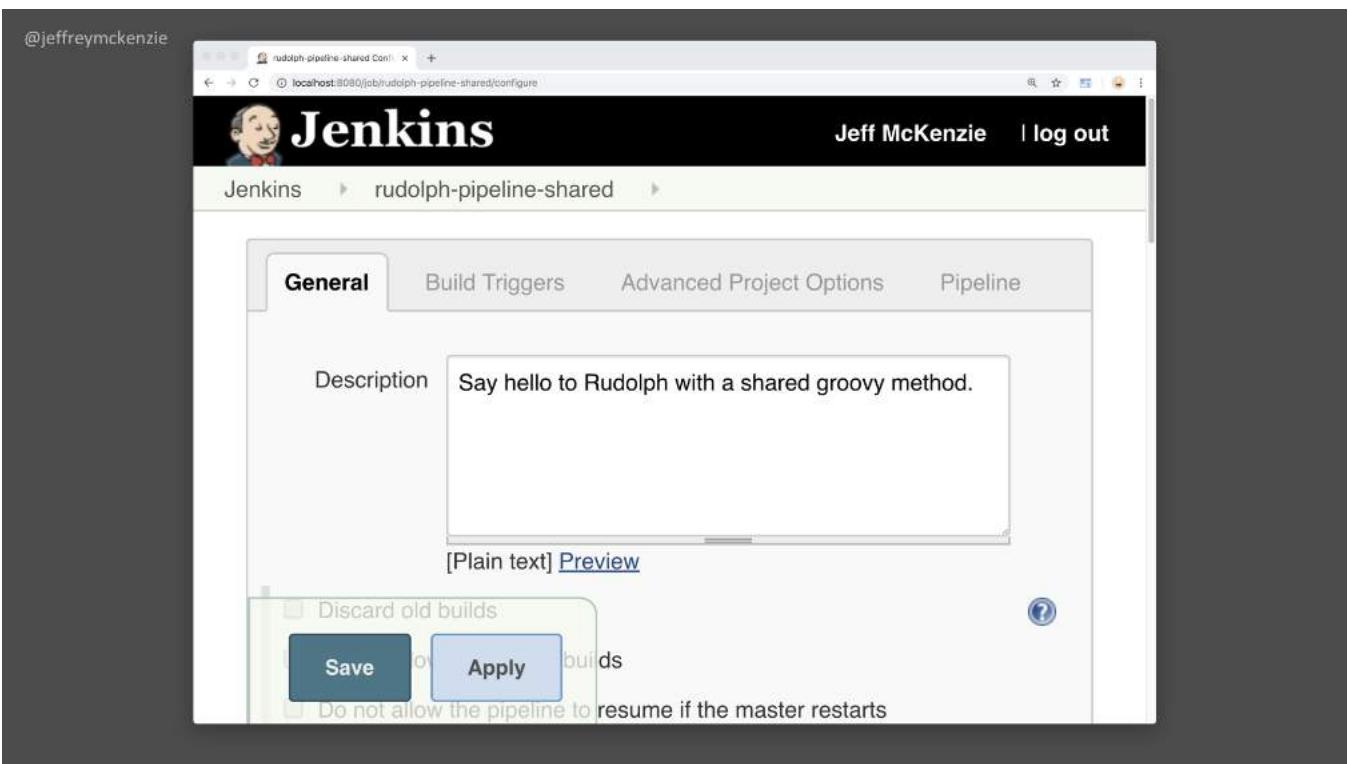
The main content area has a heading "Enter an item name" and a text input field containing "rudolph-pipeline-shared". A note below the input field says "» Required field".

Below the input field, there are three project types listed:

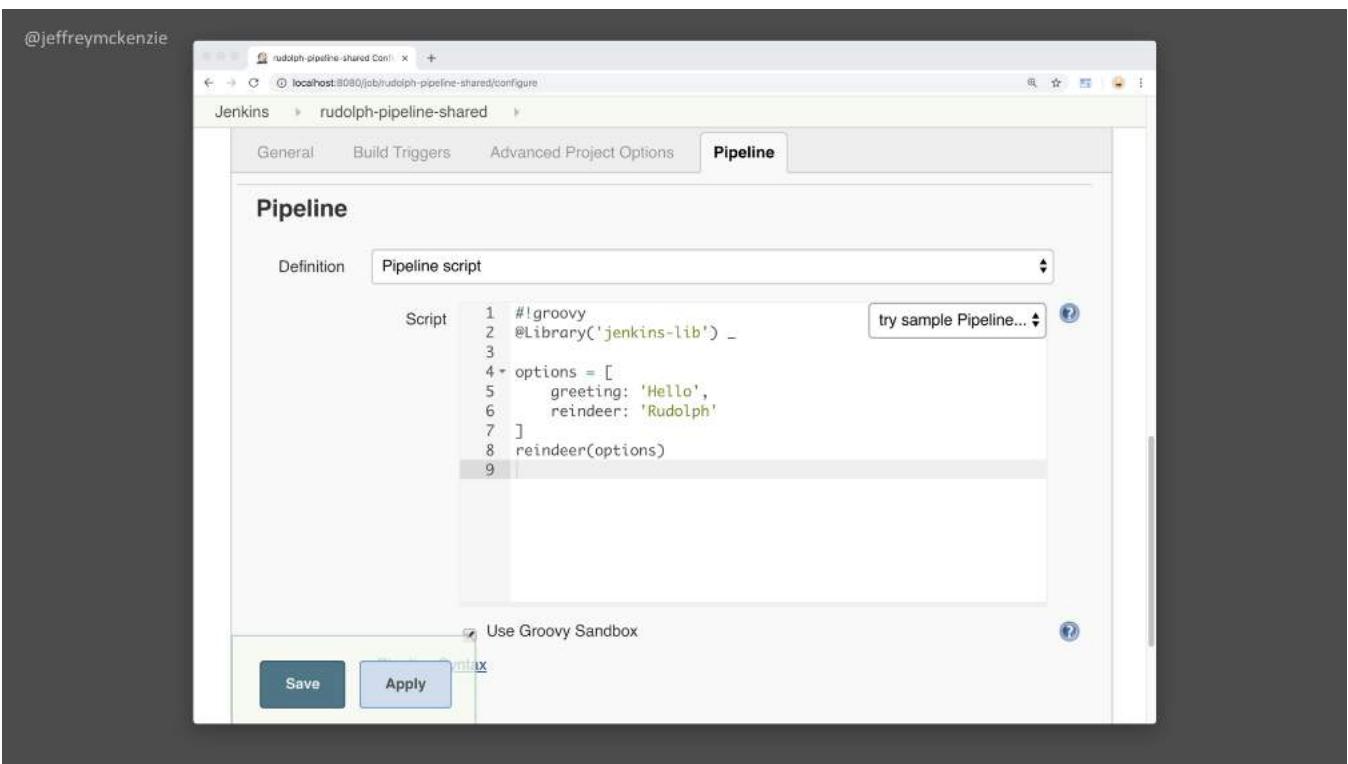
- Freestyle project**: Described as the central feature of Jenkins, combining any SCM with any build system, and even used for non-software builds. It includes a small icon of a box with a gear.
- Pipeline**: Described as orchestrating long-running activities across multiple build agents, suitable for building pipelines (formerly workflows) or organizing complex activities. It includes a small icon of a pipe.
- Multi-configuration project**: Suitable for projects needing many configurations, like testing in multiple environments. It includes a small icon of a document with a gear.

At the bottom of the form, there are two buttons: "OK" (highlighted in blue) and "Folder".

We'll call it Rudolph-pipeline-shared



We're going to say hello to Rudolph with a shared groovy method.



We go down to the pipeline script area
And just add a few lines here –

And that's it –

Lets take a closer look.

@jeffreymckenzie

```
#!groovy
@Library('jenkins-lib') _

options = [
    greeting: 'Hello',
    reindeer: 'Rudolph'
]
reindeer(options)
```

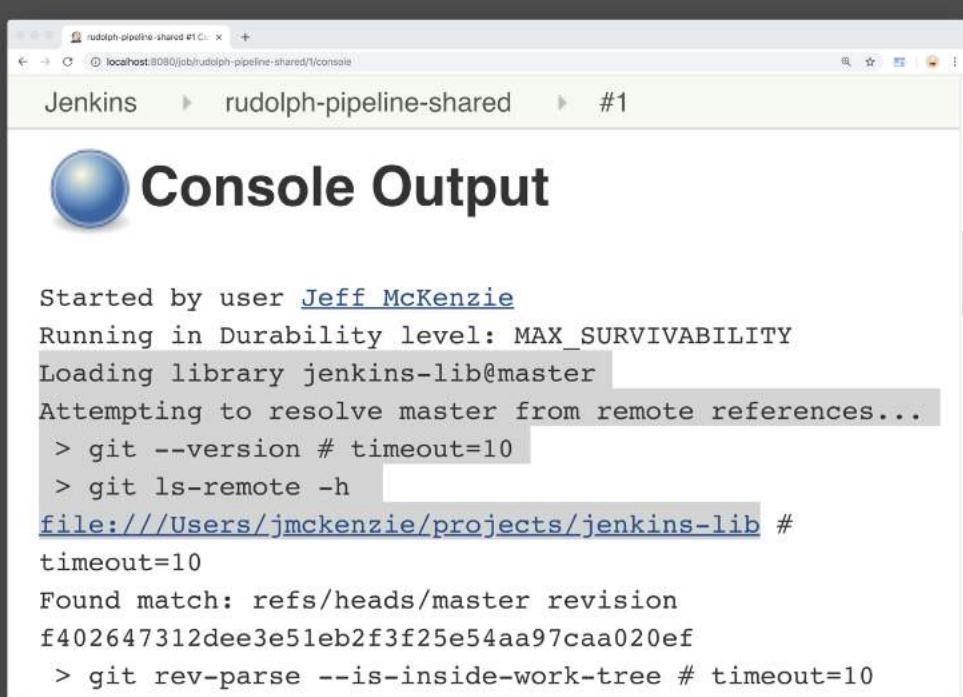
The first line tells Jenkins you're executing groovy –
Second line is a reference to the global library
You want to use – (note underscore)

Then you create the map object with the variable values,

And call the method by passing in the options.
Jenkins will look for the file called reindeer
and execute the method in it

Let's run this and look at the result...

@jeffreymckenzie

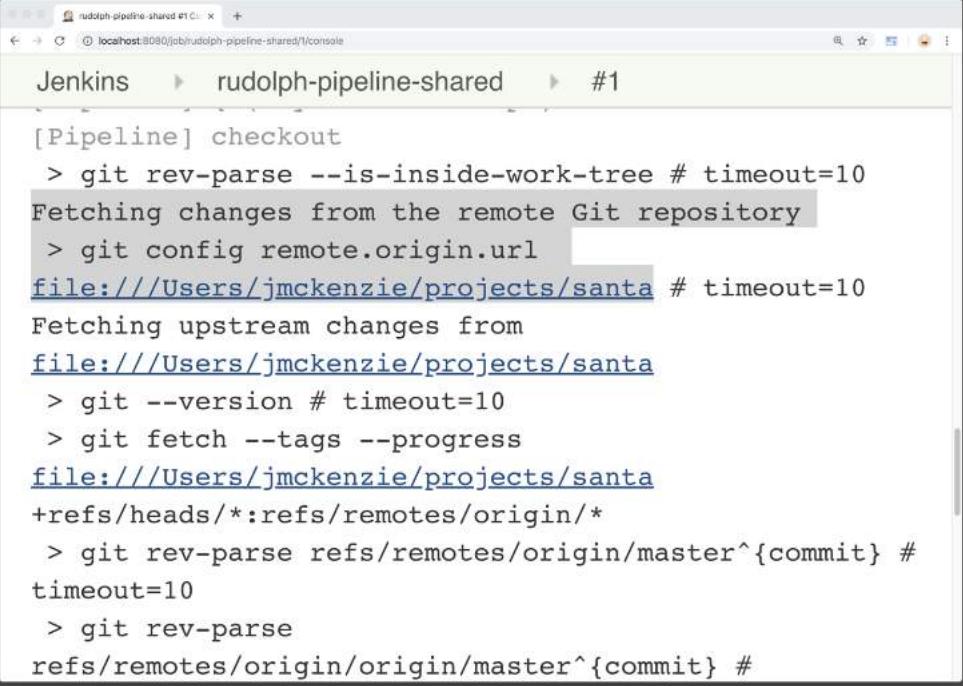


The screenshot shows a Jenkins console output window titled "Console Output". The log starts with "Started by user [Jeff McKenzie](#)". It then shows Jenkins attempting to resolve a master reference from remote references, executing git commands like "git --version", "git ls-remote -h", and "git rev-parse --is-inside-work-tree". The path "file:///Users/jmckenzie/projects/jenkins-lib" is highlighted in blue, indicating it's a link.

```
Started by user Jeff McKenzie
Running in Durability level: MAX_SURVIVABILITY
Loading library jenkins-lib@master
Attempting to resolve master from remote references...
> git --version # timeout=10
> git ls-remote -h
file:///Users/jmckenzie/projects/jenkins-lib #
timeout=10
Found match: refs/heads/master revision
f402647312dee3e51eb2f3f25e54aa97caa020ef
> git rev-parse --is-inside-work-tree # timeout=10
```

First you can see Jenkins is
loading the library from git...

@jeffreymckenzie

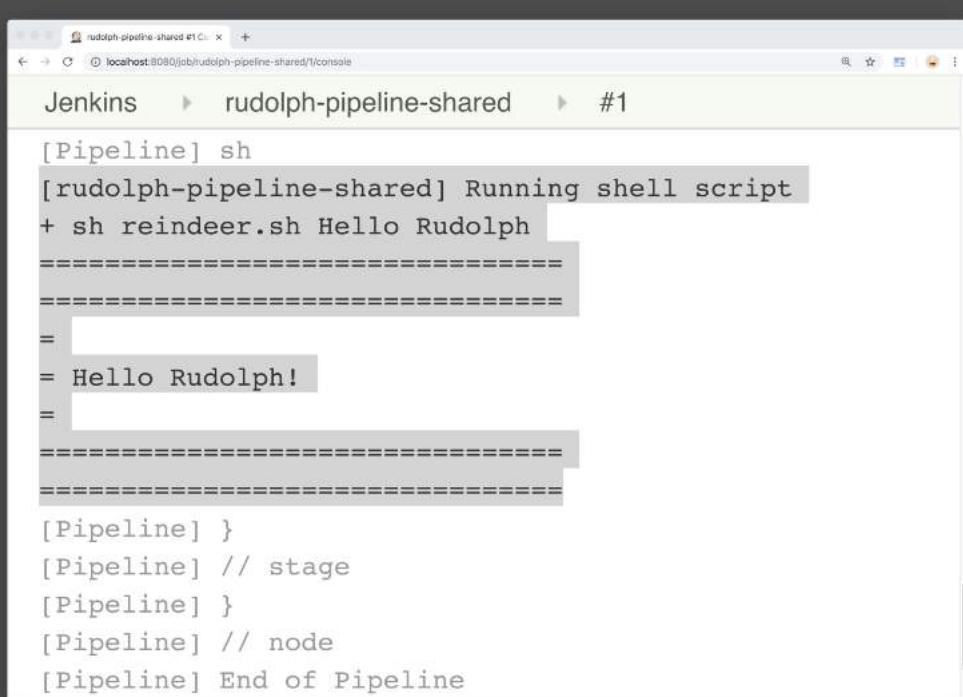


The screenshot shows a Jenkins pipeline console window titled "rudolph-pipeline-shared #1". The pipeline step being executed is "[Pipeline] checkout". The command being run is "git rev-parse --is-inside-work-tree # timeout=10". The output shows the pipeline fetching changes from a remote Git repository at "file:///Users/jmckenzie/projects/santa". It also fetches upstream changes from the same repository. The pipeline then checks for a specific commit on the master branch of the origin remote.

```
Jenkins > rudolph-pipeline-shared > #1
[Jenkins] checkout
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url
file:///Users/jmckenzie/projects/santa # timeout=10
Fetching upstream changes from
file:///Users/jmckenzie/projects/santa
> git --version # timeout=10
> git fetch --tags --progress
file:///Users/jmckenzie/projects/santa
+refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse
refs/remotes/origin/origin/master^{commit} #
```

Then it gets changes from the Santa repo...

@jeffreymckenzie

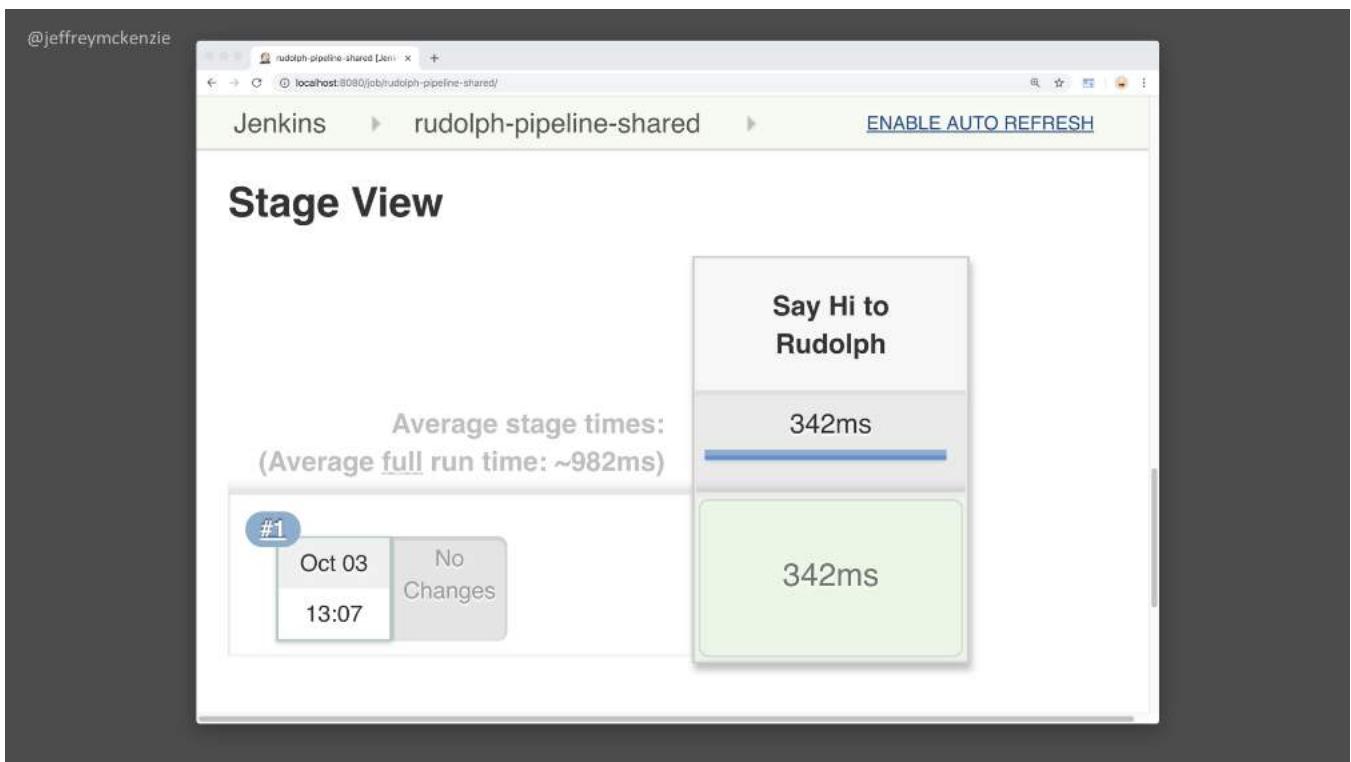


The screenshot shows a Jenkins job named "rudolph-pipeline-shared" with build number "#1". The console output is displayed in a terminal window. The output shows the execution of a pipeline script. It starts with "[Pipeline] sh", followed by "[rudolph-pipeline-shared] Running shell script". Inside the shell script block, there is a command "+ sh reindeer.sh Hello Rudolph". The output then shows the text "= Hello Rudolph!" which is the expected result of the script. The pipeline concludes with "[Pipeline] }", "[Pipeline] // stage", "[Pipeline] }", "[Pipeline] // node", and "[Pipeline] End of Pipeline".

```
[Pipeline] sh
[rudolph-pipeline-shared] Running shell script
+ sh reindeer.sh Hello Rudolph
=====
=====
=
= Hello Rudolph!
=
=====
=====
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
```

Then it runs our script
and we get the expected output...

@jeffreymckenzie



And when we look at the stage view
We can see our variable value (Rudolph)
Was used to name it.

@jeffreymckenzie



So that's better – we have a very small project/job
That uses shared code and is easy to maintain.

However, we still have to create all those jobs.

Now Santa's a busy man...

=====

<https://commons.wikimedia.org/wiki/File:HanRenne.JPG>
By GrottesdeHan [CC BY-SA 3.0
(<https://creativecommons.org/licenses/by-sa/3.0/>), from Wikimedia Commons

@jeffreymckenzie



He and Mrs claus have a lot to do around Christmas,
And he really doesn't want to have to create millions of individual jobs.

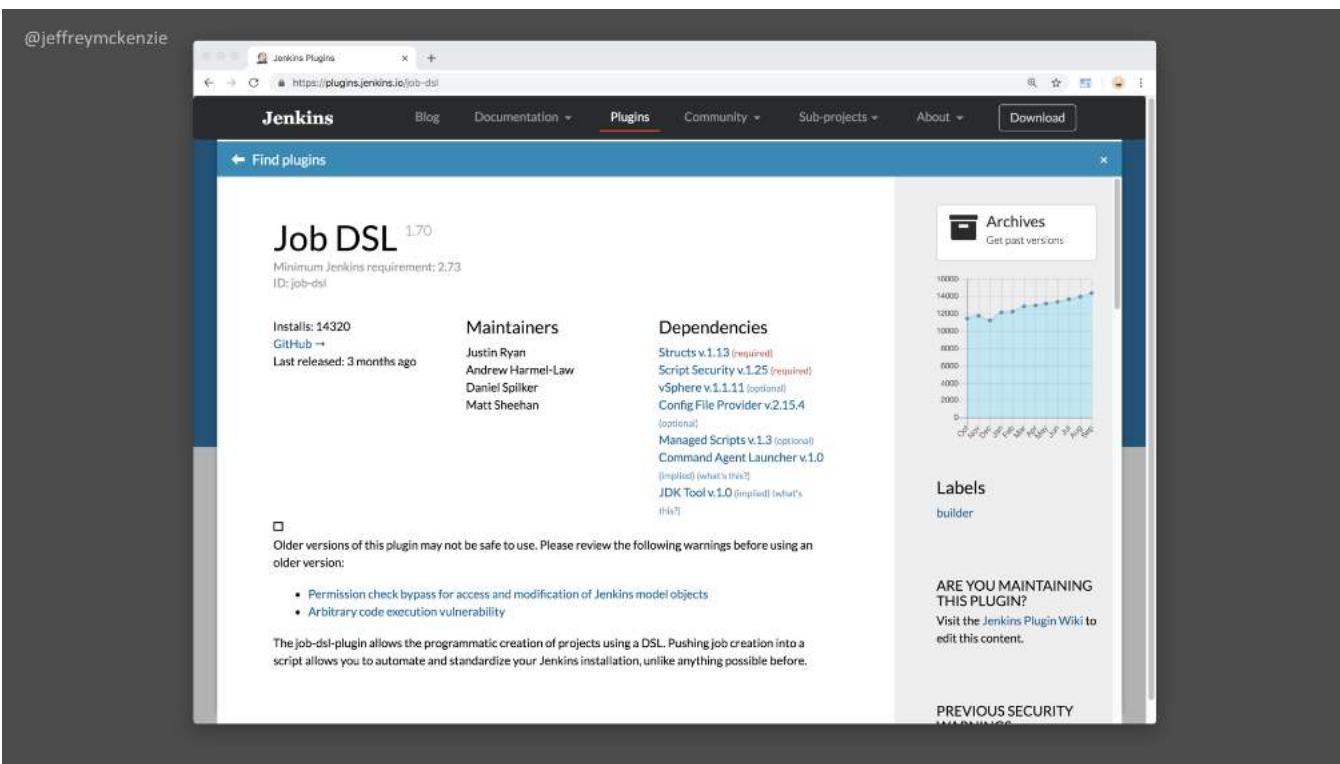
He's already automated his communication with his reindeer –

Wouldn't it be nice if he could automate his automation?

Well it turns out he can...

https://commons.wikimedia.org/wiki/File:Christmas_came_early_for_one_Alaska_village_151016-Z-MW427-433.jpg

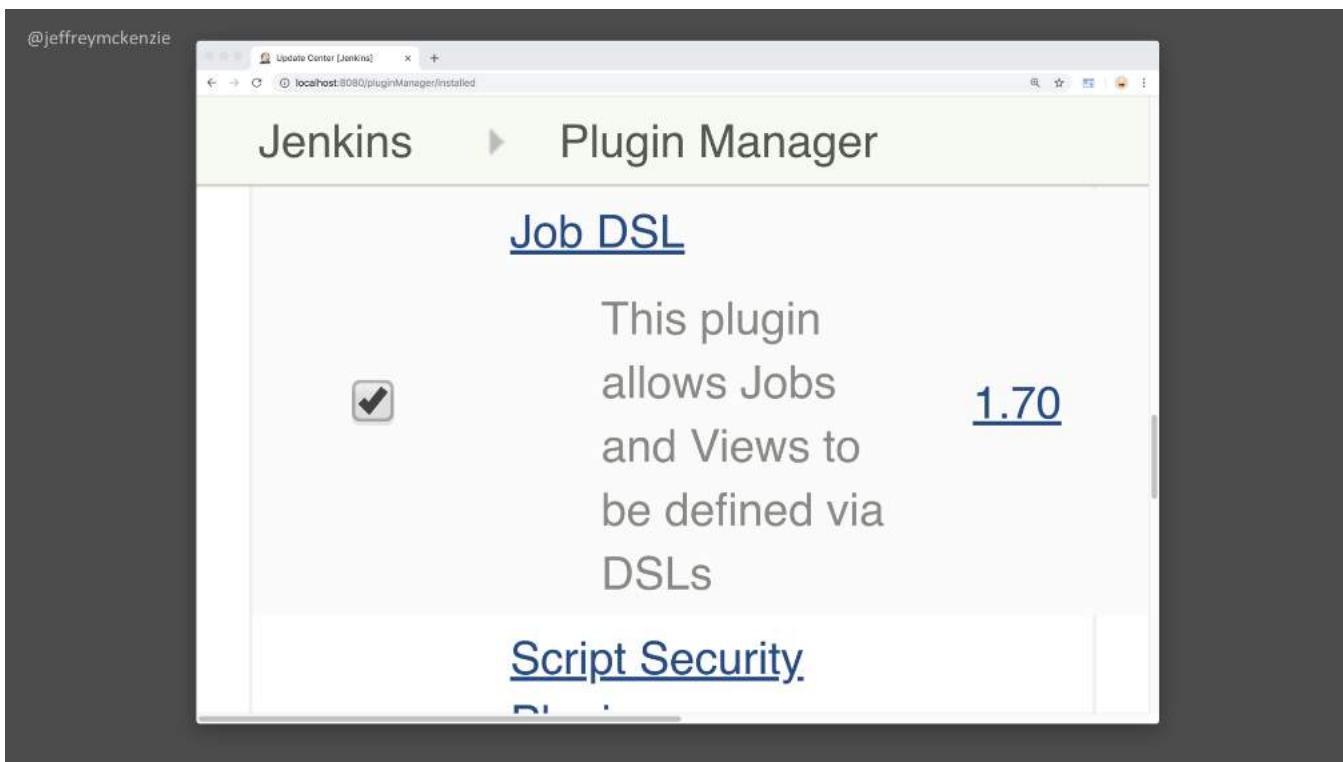
By Staff Sgt. Edward Eagerton [Public domain], via Wikimedia Commons



Using the Job DSL plugin for Jenkins.

- a pipeline script allows you to
Programmatically control what a job does,
- A job dsl script allows you to
Programmatically create jobs themselves.

@jeffreymckenzie



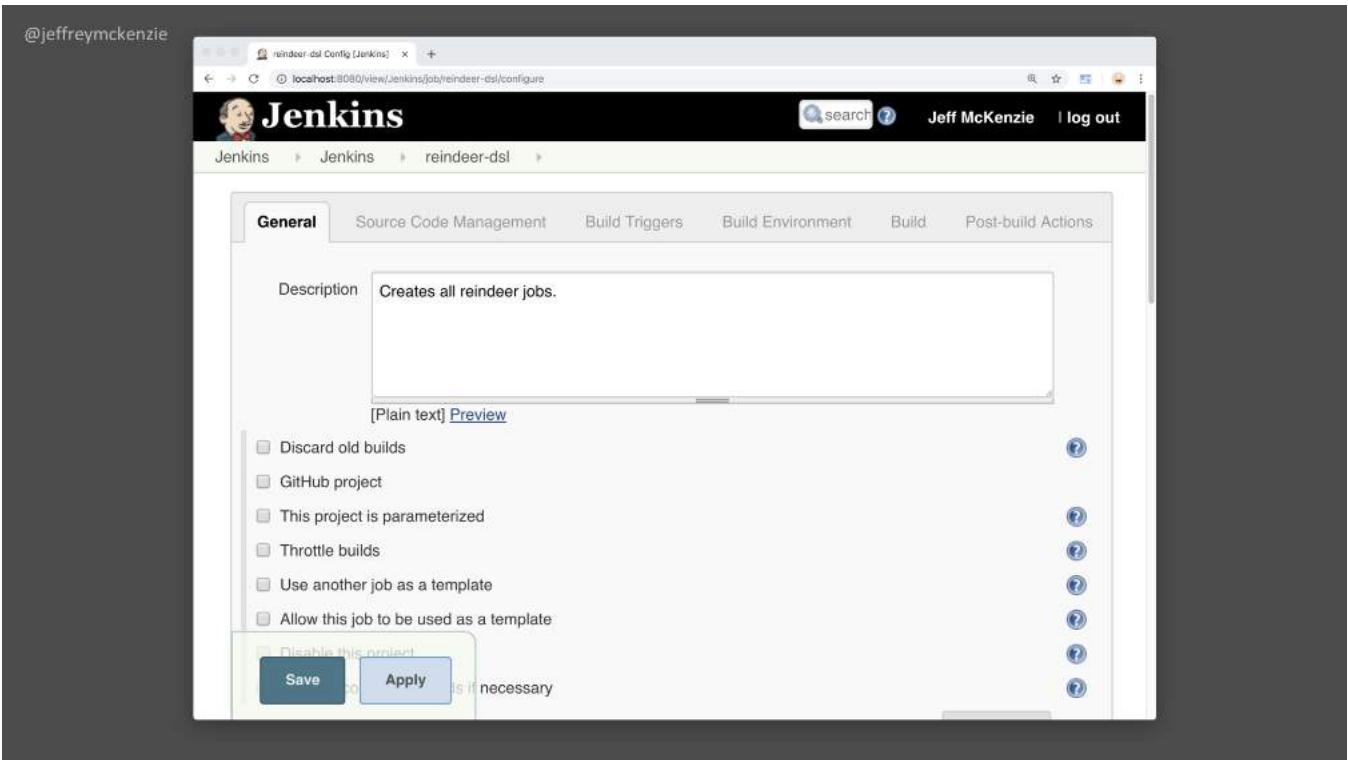
This is not built into Jenkins,
But you can install it through plugin Manager.

@jeffreymckenzie

The screenshot shows the Jenkins web interface for creating a new item. At the top, there's a header with the Jenkins logo, the user name 'Jeff McKenzie', and a 'log out' link. Below the header, the title 'Enter an item name' is displayed, followed by a text input field containing 'reindeer-dsl'. A note below the input field says '» Required field'. There are three main project types listed: 'Freestyle project' (selected), 'Pipeline', and 'multi-configuration project'. Each type has a brief description and a small icon. A blue 'OK' button is visible at the bottom left of the list.

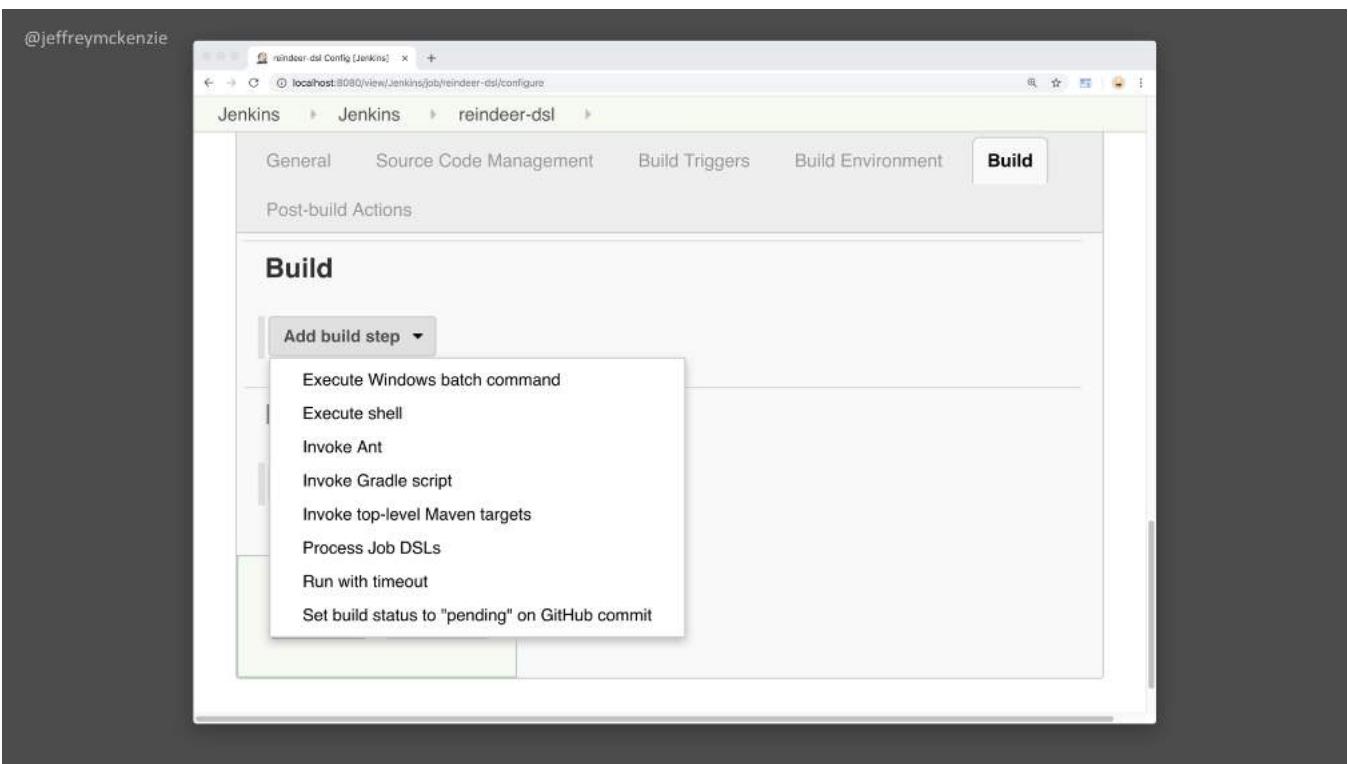
Let's see how that works --

You create a DSL job as a freestyle project



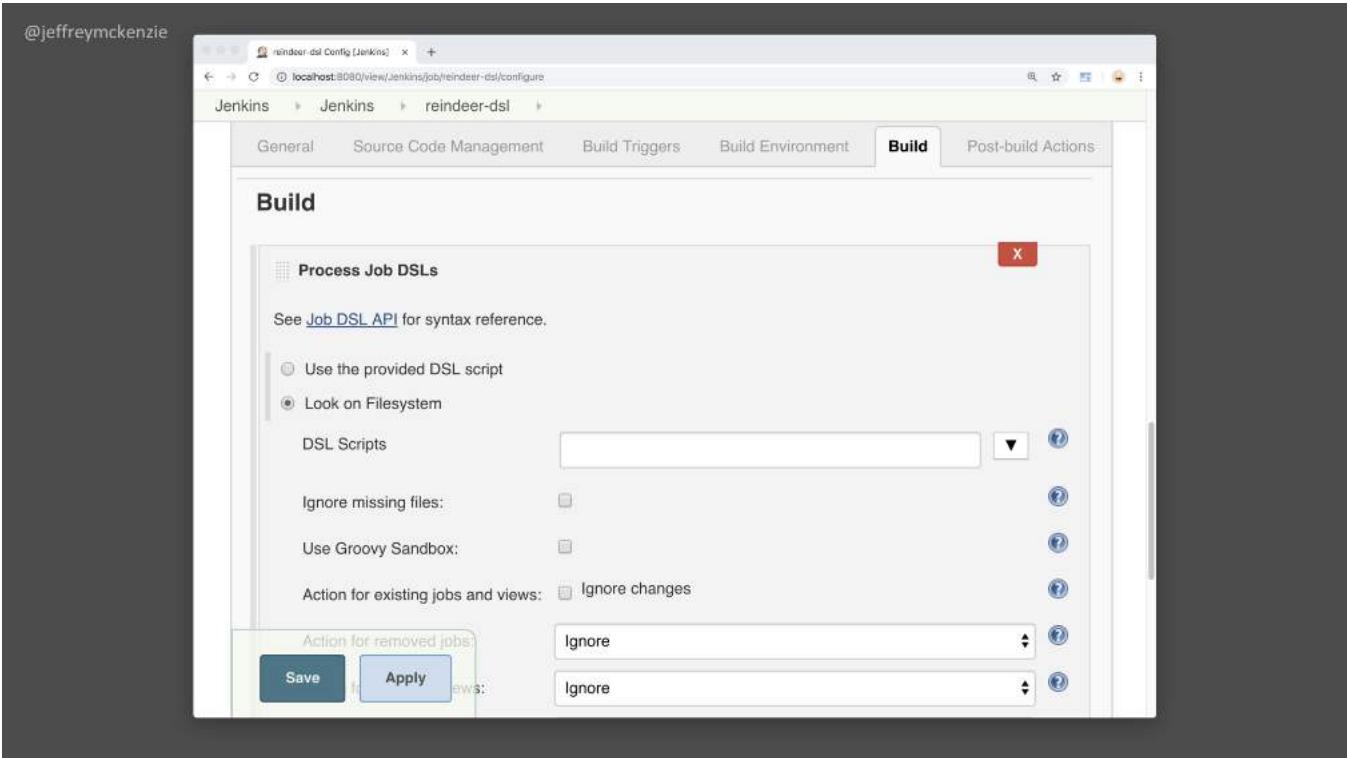
Give description –

Creates all reindeer jobs



Then we go down to build and select the step
Process job DSLs –

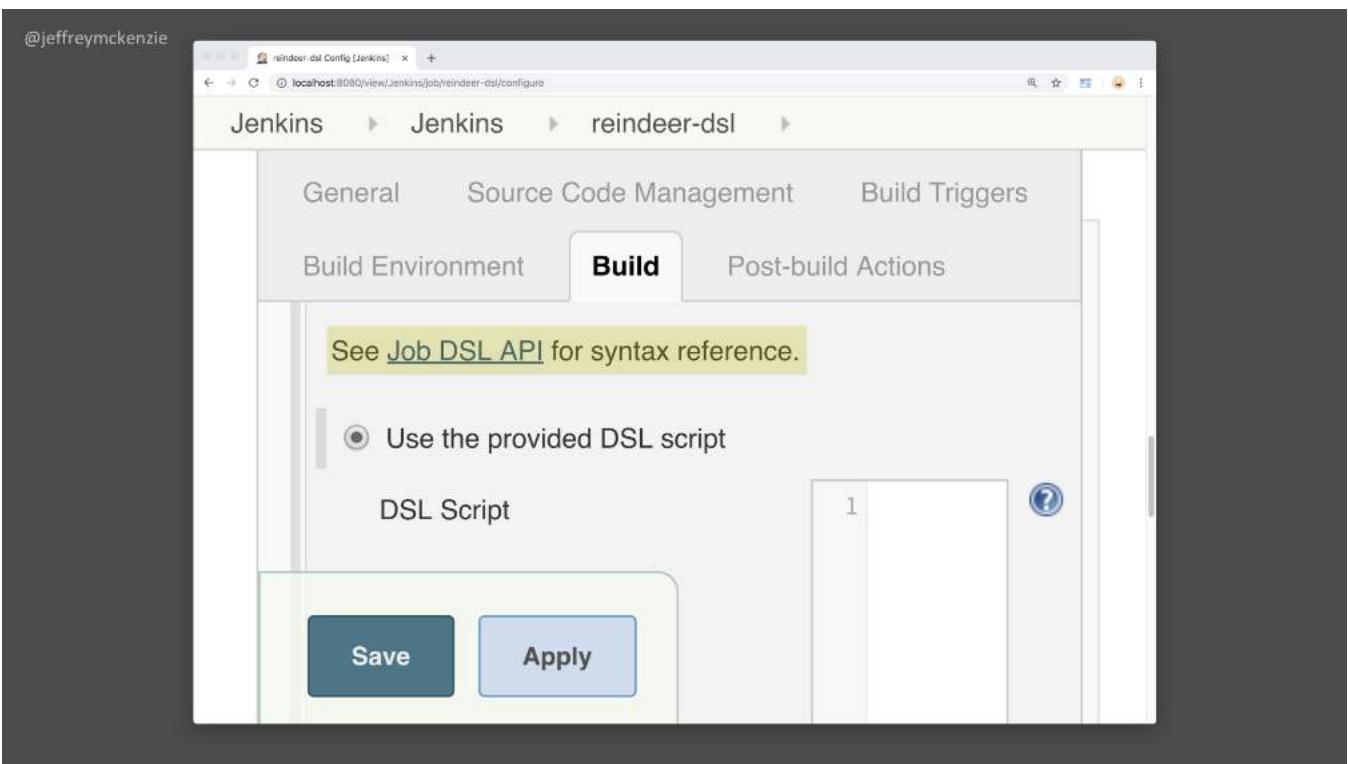
Again, this will only show up
if you have the plugin installed



Have a couple options –

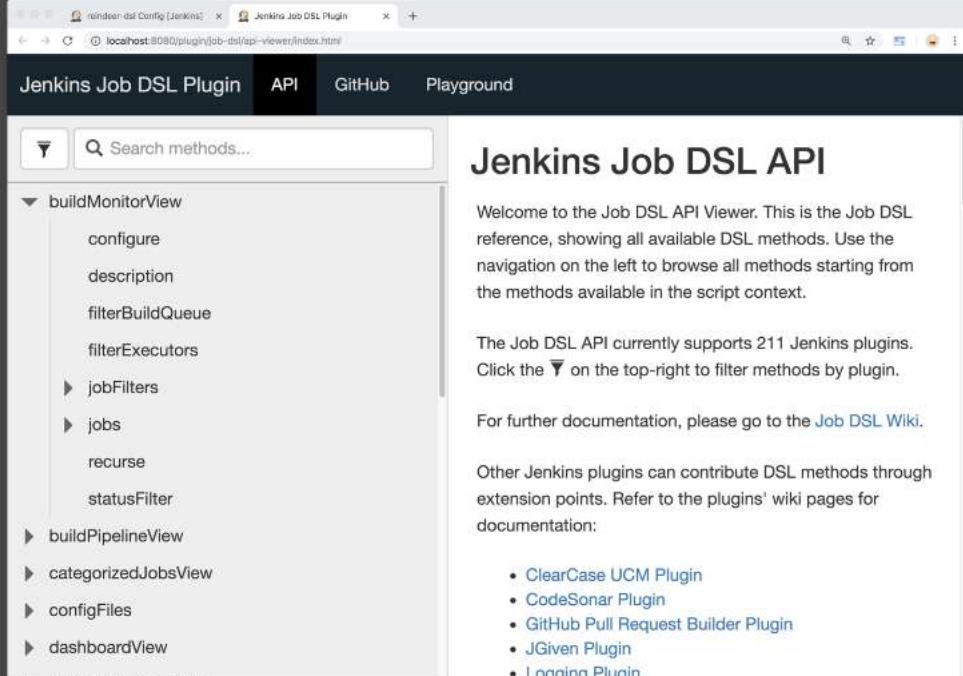
We can pull from filesystem,
Which means we can version control this –

Or we can enter a script right here,
Which we will do for demo purposes.



And on that page there's a link to the
Job DSL API that you can visit
To learn more about the scripting syntax

@jeffreymckenzie



The screenshot shows a browser window titled "Jenkins Job DSL Plugin" with the URL "localhost:8080/plugin/job-dsl/api-viewer/index.html". The page has a dark header with tabs for "Jenkins Job DSL Plugin", "API", "GitHub", and "Playground". On the left is a sidebar with a search bar and a tree view of available methods under "buildMonitorView". The main content area is titled "Jenkins Job DSL API" and contains a welcome message, information about supported plugins, links to documentation, and a list of other contributing plugins.

Jenkins Job DSL API

Welcome to the Job DSL API Viewer. This is the Job DSL reference, showing all available DSL methods. Use the navigation on the left to browse all methods starting from the methods available in the script context.

The Job DSL API currently supports 211 Jenkins plugins. Click the ▼ on the top-right to filter methods by plugin.

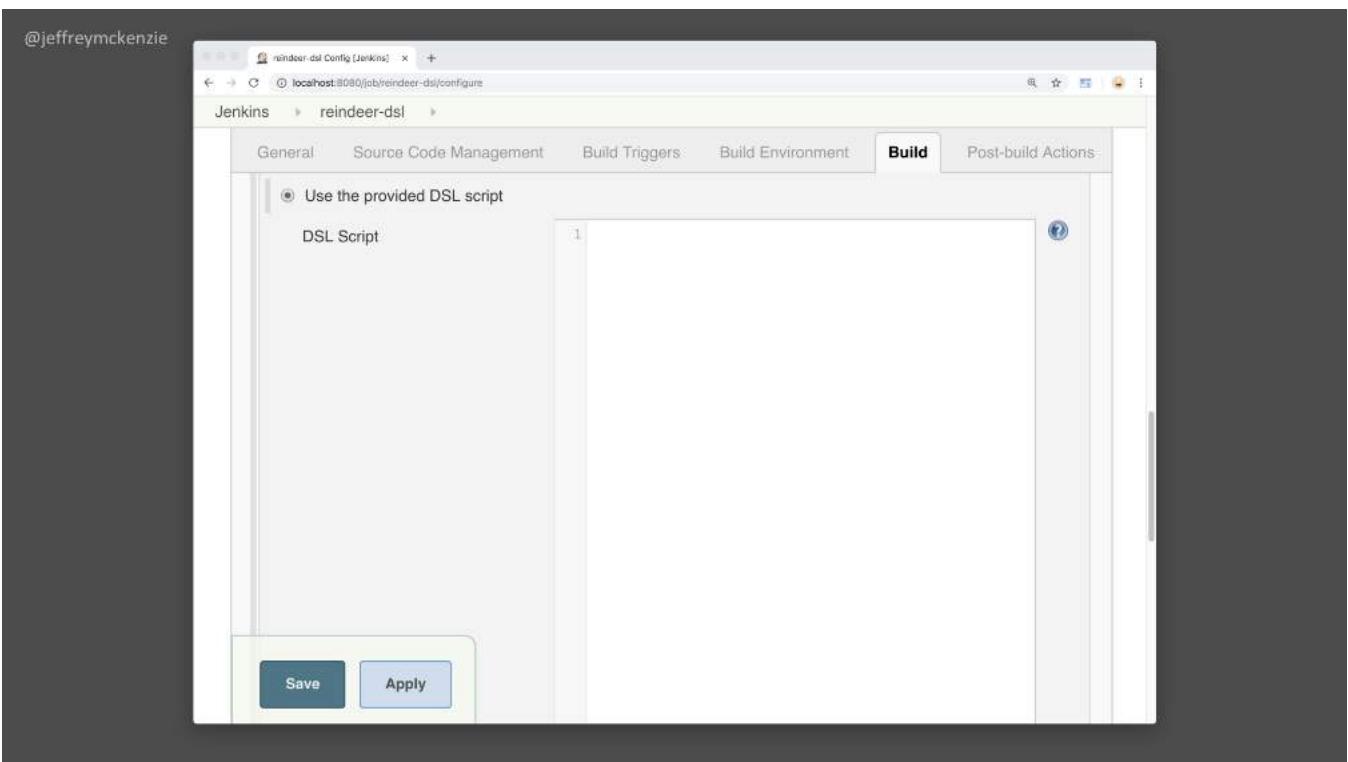
For further documentation, please go to the [Job DSL Wiki](#).

Other Jenkins plugins can contribute DSL methods through extension points. Refer to the plugins' wiki pages for documentation:

- ClearCase UCM Plugin
- CodeSonar Plugin
- GitHub Pull Request Builder Plugin
- JGiven Plugin
- Logging Plugin

It has a really nice viewer
That you can search
To find out more about how
To create Jenkins items
Through the DSL

@jeffreymckenzie



So here's the space to put our script
Before we do this,
We need to make one adjustment to our pipeline script....

@jeffreymckenzie

```
#!groovy
@Library('jenkins-lib') _

options = [
    greeting: 'Hello',
    reindeer: 'Rudolph'
]
reindeer(options)
```

Going to do 2 things –

1. Put this in source control under the name “Jenkinsfile”
2. Switch out the hard-coded values to use parameters

@jeffreymckenzie

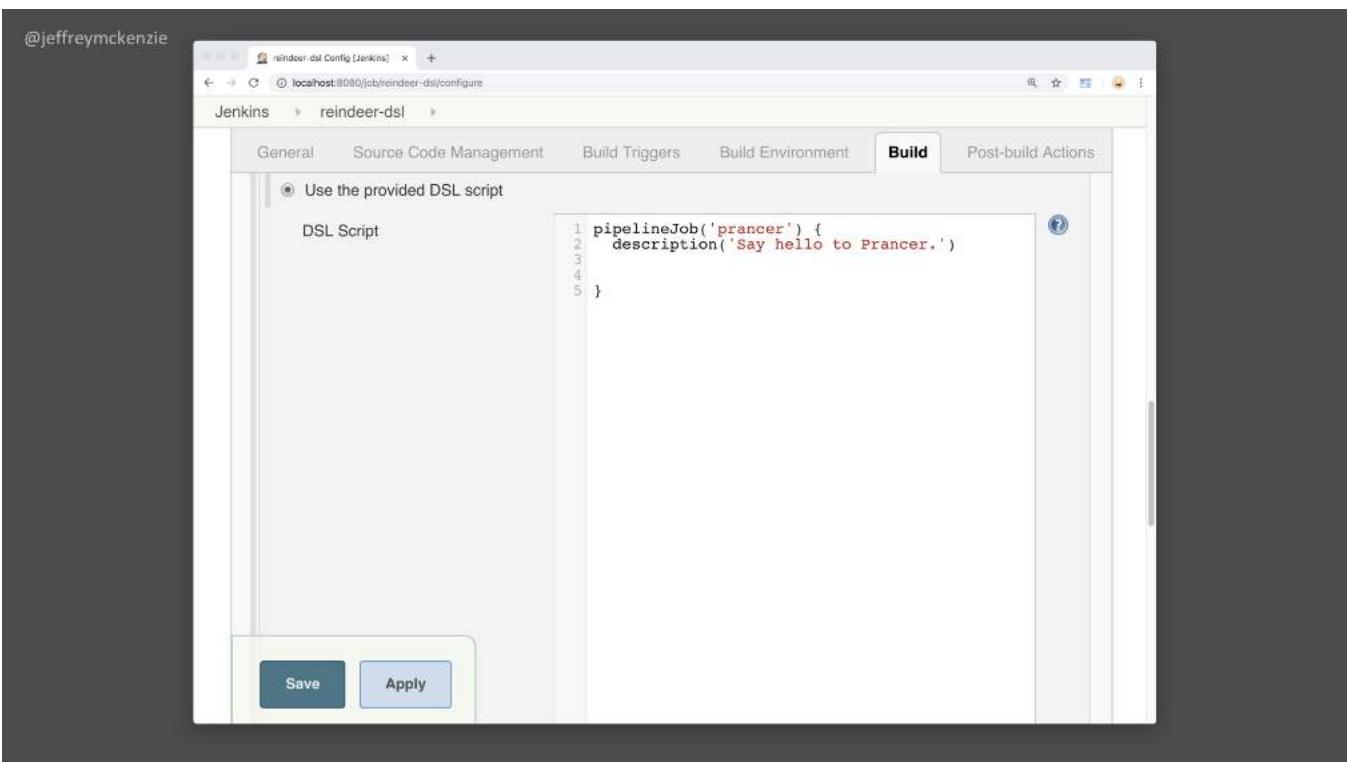
```
- Jenkinsfile
#!groovy
@Library('jenkins-lib') _

options = [
    greeting: GREETING,
    reindeer: REINDEER
]
reindeer(options)
```

Otherwise, it's the same.

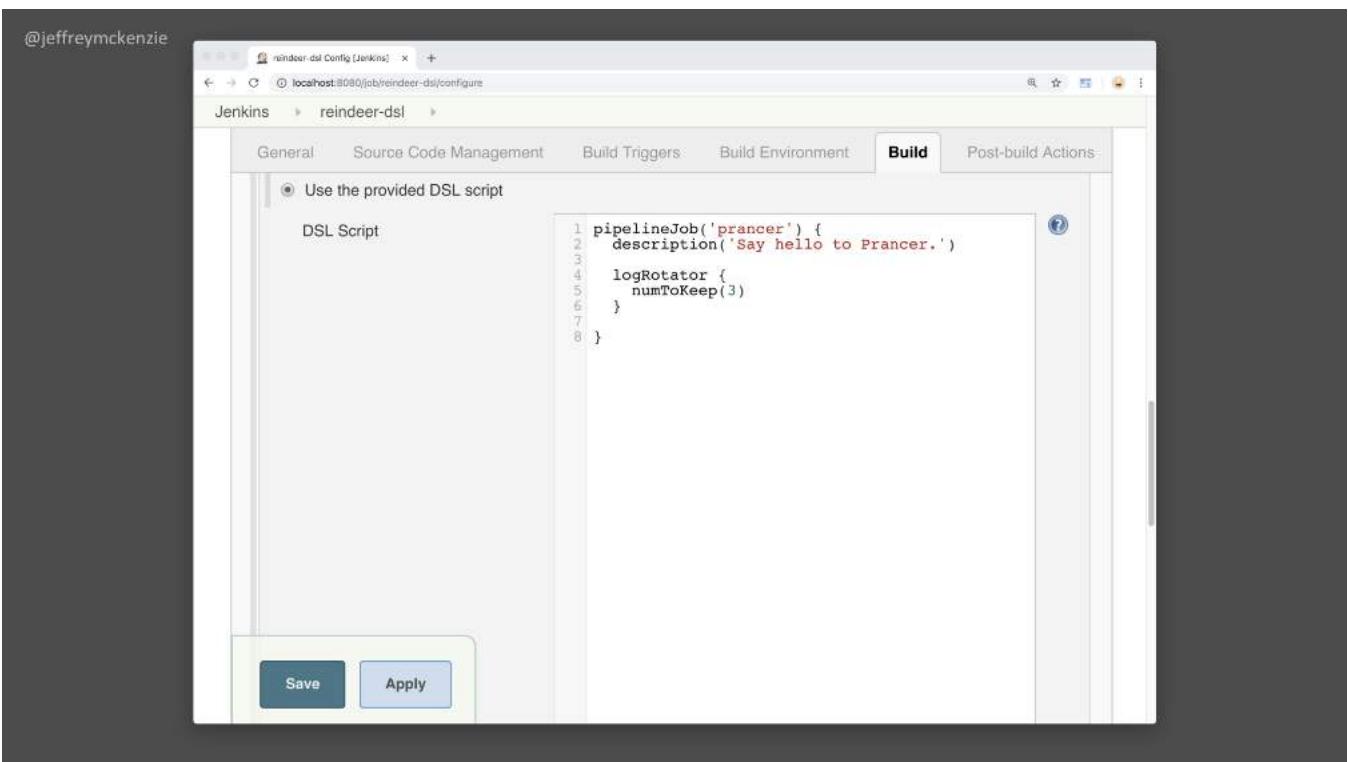
Lets go back to our dsl project and start to build it

@jeffreymckenzie



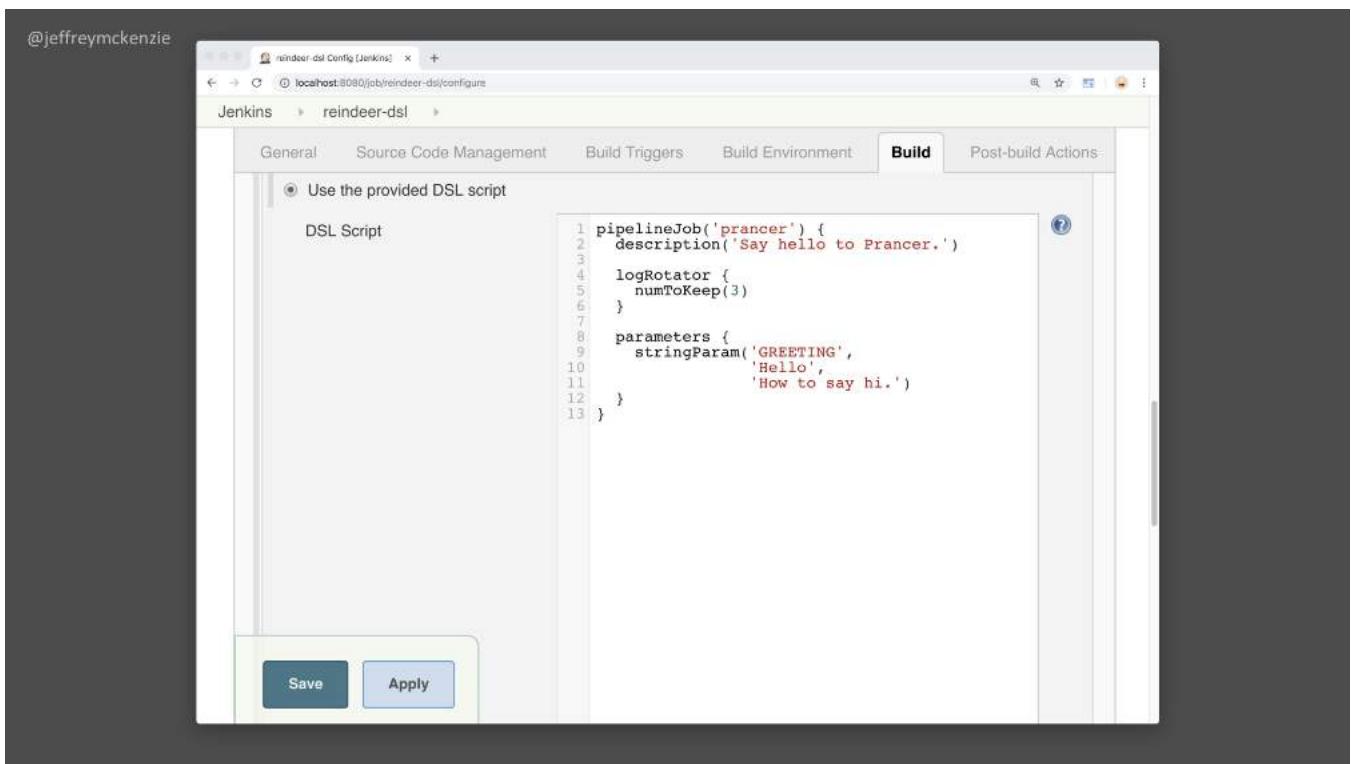
To create a pipeline job or project,
We simply use the pipelineJob command,
With the name of the project –
We'll call this one Prancer.
Then we give it a description.

@jeffreymckenzie



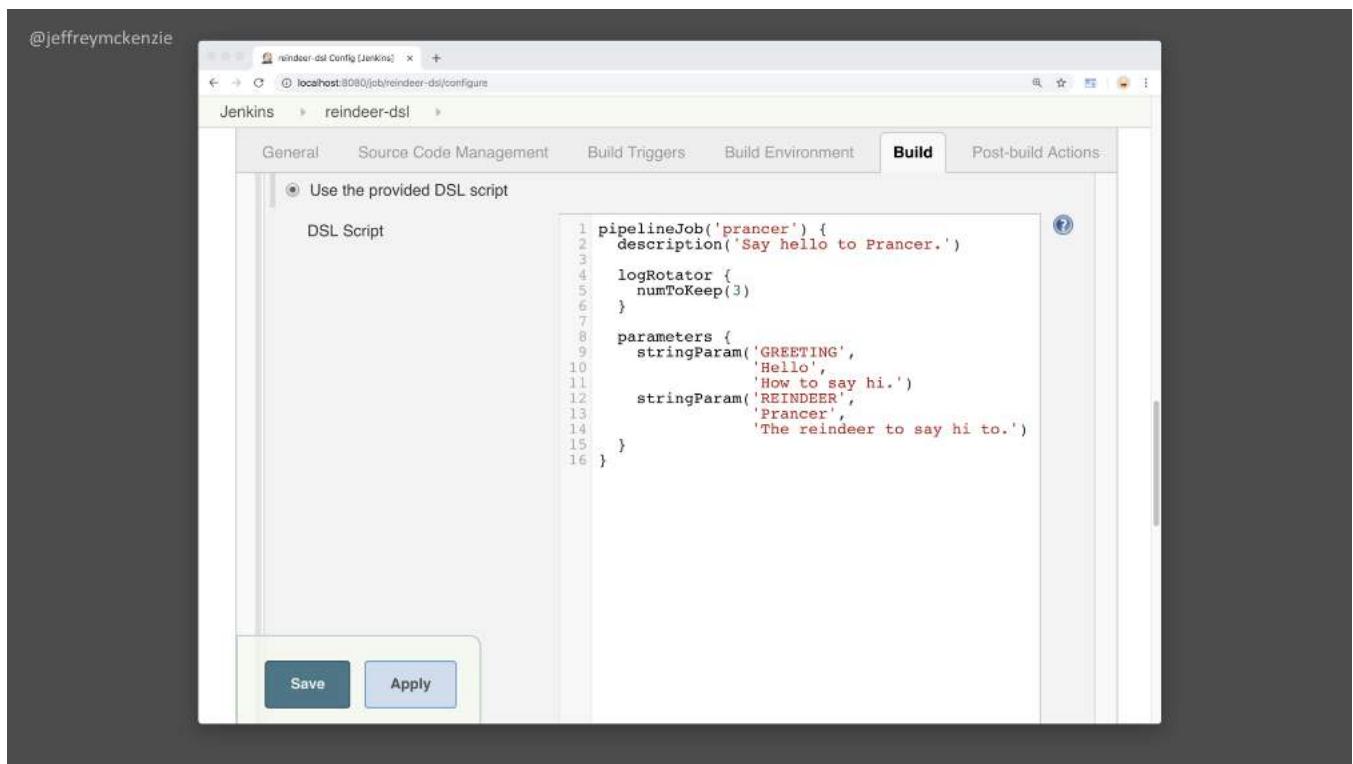
We add our build/log history –
We want to keep the last 3 build logs.

@jeffreymckenzie



We add our GREETING parameter

@jeffreymckenzie



Then our reindeer parameter

@jeffreymckenzie

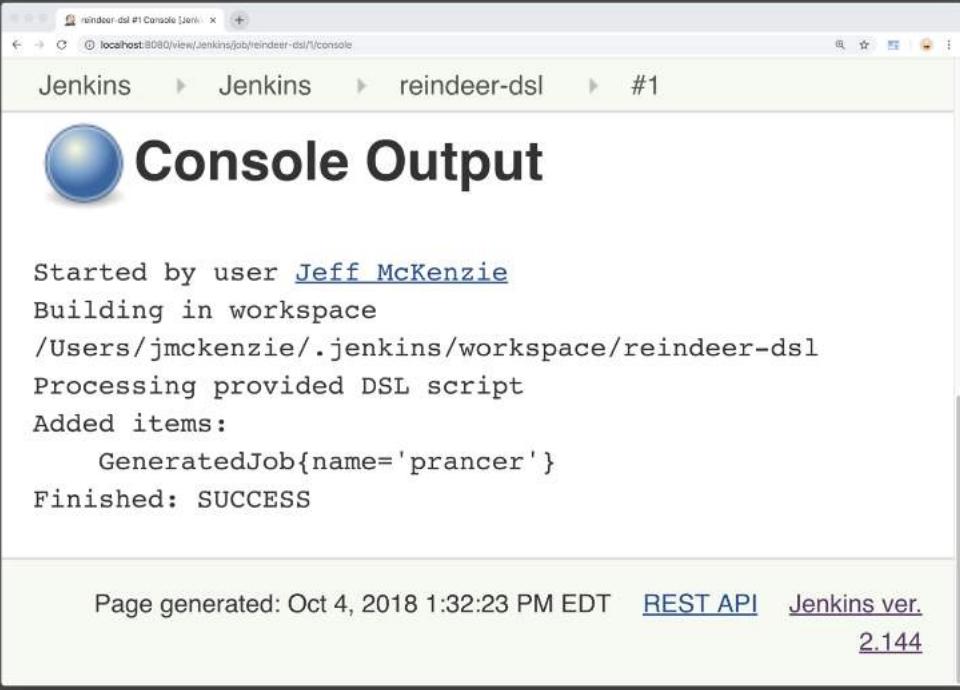
The screenshot shows the Jenkins configuration interface for a job named 'reindeer-dsl'. The 'Build' tab is selected. Under 'Use the provided DSL script', the following code is displayed:

```
1 pipelineJob('prancer') {
2     description('Say hello to Prancer.')
3 
4     logRotator {
5         numToKeep(3)
6     }
7 
8     parameters {
9         stringParam('GREETING',
10            'Hello',
11            'How to say hi.')
12         stringParam('REINDEER',
13            'Prancer',
14            'The reindeer to say hi to.')
15     }
16 
17     definition {
18         cpsScm {
19             scm {
20                 git {
21                     branch('*/*')
22                     remote {
23                         url(
24                             'file:///Users/jmckenzie/projects/santa')
25                         }
26                     }
27                     scriptPath('Jenkinsfile')
28                 }
29             }
30         }
31     }
}
```

At the bottom left are 'Save' and 'Apply' buttons.

And then finally the build definition itself –
This just tells Jenkins what pipeline script to pull
And which git repo to get it from.

So let's run this and see what it does...

```
@jeffreymckenzie


Started by user Jeff McKenzie  
Building in workspace  
/Users/jmckenzie/.jenkins/workspace/reindeer-dsl  
Processing provided DSL script  
Added items:  
    GeneratedJob{name='prancer'}  
Finished: SUCCESS



Page generated: Oct 4, 2018 1:32:23 PM EDT    REST API    Jenkins ver.  
2.144

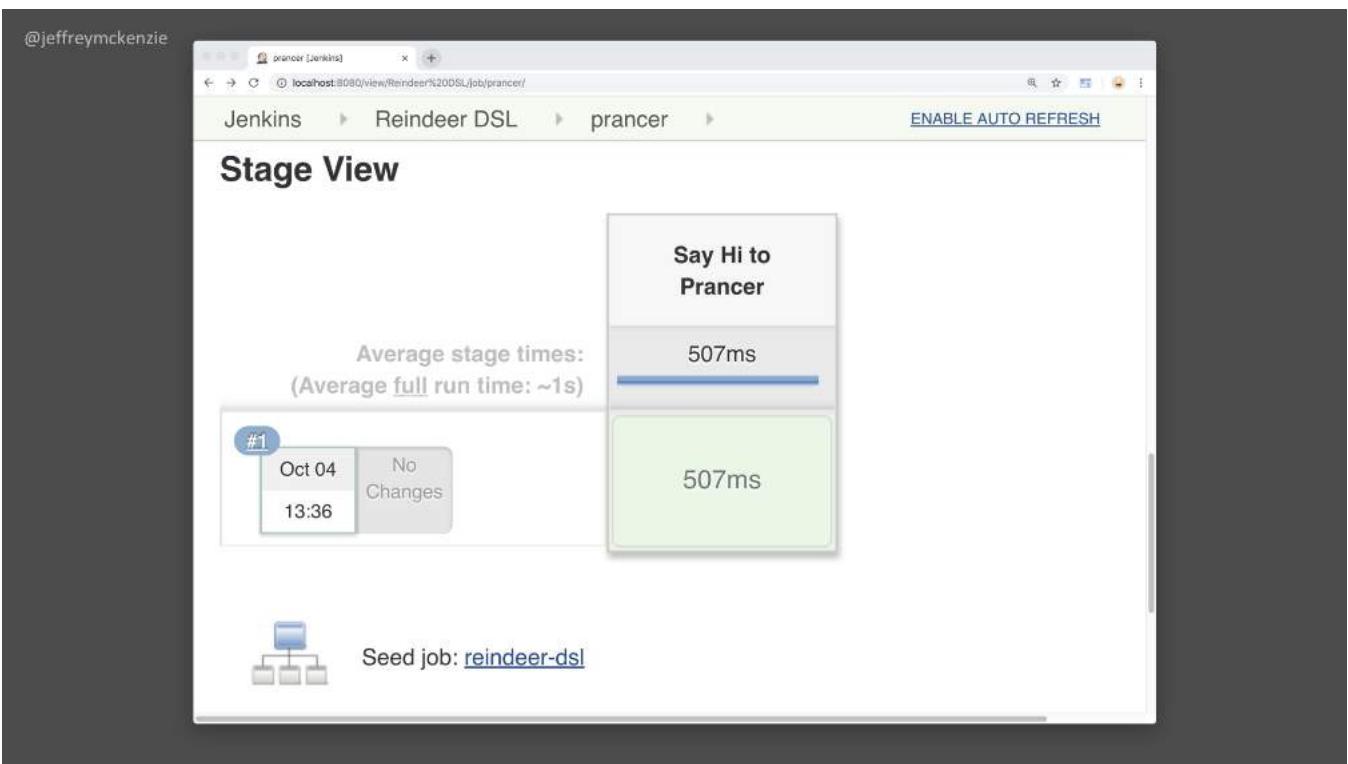

```

So it tells us we've created
the prancer job successfully..
Let's go take a look.

@jeffreymckenzie

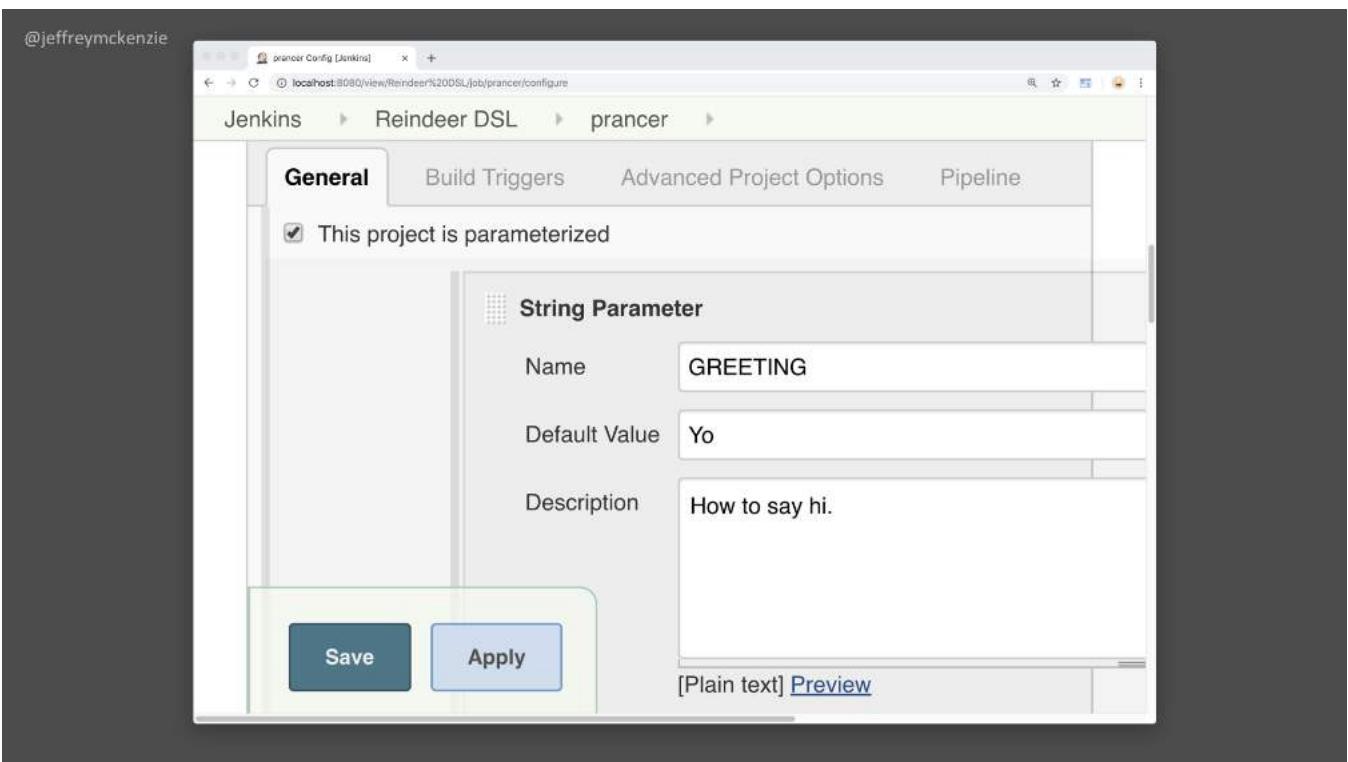
S	W	Name ↓	Last Success	Last Failure	Last Duration
Grey	Yellow	orancer	N/A	N/A	N/A
Blue	Yellow	reindeer-dsl	2 min 42 sec - #1	N/A	0.19 sec

So if we go back to the main list,
You can see the prancer job,
Which was created programmatically
Through the DSL Script.
Let's run it.



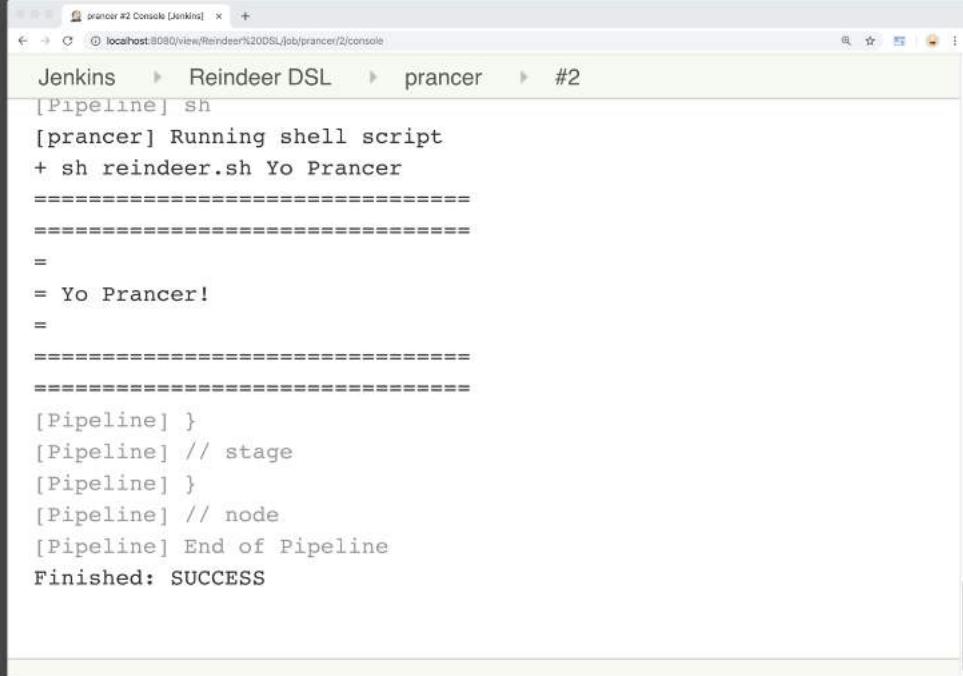
If we look at the stage view,
we can see that it runs successfully,
But this time saying hi to prancer--

Also at the bottom you can see that
The DSL seed job is listed below.



Now if we go in and manually change this job,
Let's so we want to change to say Yo to prance,
We can do so.

@jeffreymckenzie



The screenshot shows a Jenkins console output window titled "prancer #2 Console [Jenkins]". The URL in the address bar is "localhost:8080/view/Reindeer%20DSL/job/prancer/2/console". The console output is as follows:

```
Jenkins   ▶ Reindeer DSL   ▶ prancer   ▶ #2
[Pipeline] sh
[prancer] Running shell script
+ sh reindeer.sh Yo Prancer
=====
=====
=
= Yo Prancer!
=
=====
=====
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

And it will run,
But if we go back

@jeffreymckenzie

Jenkins → Reindeer DSL → prancer → [ENABLE AUTO REFRESH](#)

Oct 04
13:36
No Changes
507ms

Seed job: [reindeer-dsl](#)

⚠ This item has been changed manually since it was generated by the seed job.

Permalinks

- [Last build \(#2\), 1 min 34 sec ago](#)
- [Last stable build \(#2\), 1 min 34 sec ago](#)
- [Last successful build \(#2\), 1 min 34 sec ago](#)
- [Last completed build \(#2\), 1 min 34 sec ago](#)

It shows us that it's been manually changed,
Which is great so we know if
Someone has messed with since it was generated.

So this is all good, but we've only created one job.

The power of Job DSL is that it allows you
To add scripting to declaration

@jeffreymckenzie

```
1 pipelineJob('prancer') {
2     description('Say hello to Prancer.')
3 
4     logRotator {
5         numToKeep(3)
6     }
7 
8     parameters {
9         stringParam('GREETING',
10            'Hello',
11            'How to say hi.')
12         stringParam('REINDEER',
13            'Prancer',
14            'The reindeer to say hi to.')
15     }
16 
17     definition {
18         cpsScm {
19             scm {
20                 git {
21                     branch('*/*')
22                     remote {
23                         url(
24                             'file:///Users/jmckenzie/projects/santa')
25                     }
26                     scriptPath('Jenkinsfile')
27                 }
28             }
29         }
30     }
31 }
```

So here's our original DSL script –
Creates the prancer job.

Let's modify that slightly.

@jeffreymckenzie

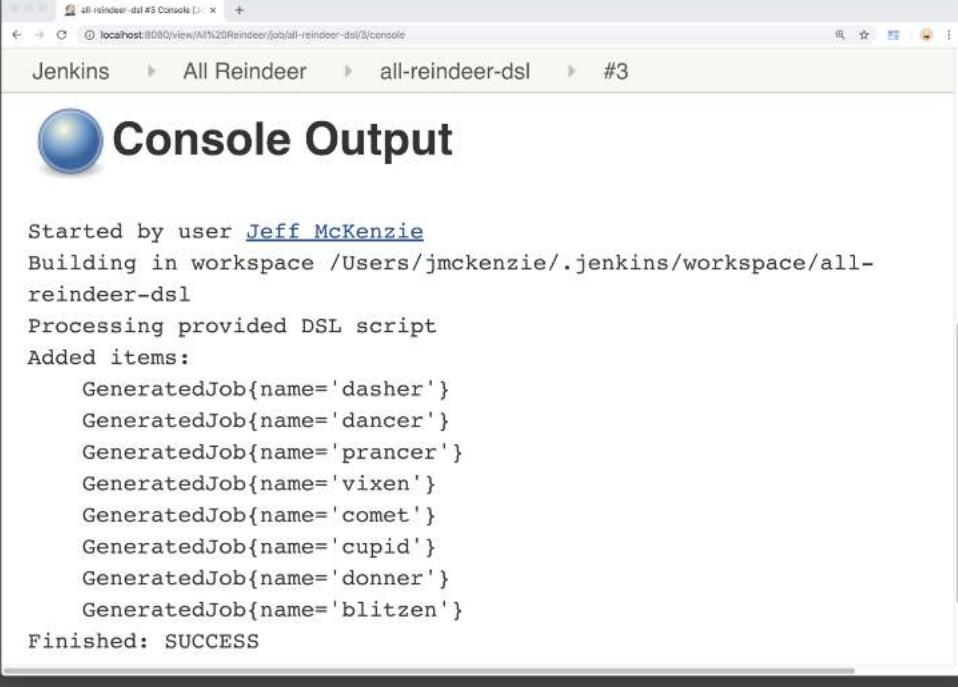
The screenshot shows the Jenkins configuration interface for a job named 'all-reindeer-dsl'. The 'Build' tab is selected. In the 'DSL Script' section, the following Groovy code is displayed:

```
names = ['dasher', 'dancer', 'prancer', 'vixen',  
        'comet', 'cupid', 'donner', 'blitzen']  
  
names.each { n ->  
    String name = n  
  
    pipelineJob(name) {  
        description('Say hello to ' + name + '.')  
        logRotator {  
            numToKeep(3)  
        }  
        parameters {  
            stringParam('GREETING', 'Hello',  
                       'How to say hi.')  
            stringParam('REINDEER',  
                       name,  
                       'The reindeer to say hi to.')  
        }  
        definition {  
            cpsScm {  
                scm {  
                    git {  
                        branch('*/*master')  
                        remote {  
                            url('file:///Users/jmckenzie/projects/santa')  
                        }  
                        scriptPath('Jenkinsfile')  
                    }  
                }  
            }  
        }  
    }  
}
```

At the bottom left of the code editor, there are two buttons: 'Save' and 'Apply'.

At the top we are going to add an array
Of reindeer names,
Loop through each of those names,
And create a job for each name,
Inserting the name where we need it.
Let's run this.

@jeffreymckenzie



The screenshot shows a Jenkins console output window titled "Console Output". The output text is as follows:

```
Started by user Jeff McKenzie
Building in workspace /Users/jmckenzie/.jenkins/workspace/all-reindeer-dsl
Processing provided DSL script
Added items:
    GeneratedJob{name='dasher'}
    GeneratedJob{name='dancer'}
    GeneratedJob{name='prancer'}
    GeneratedJob{name='vixen'}
    GeneratedJob{name='comet'}
    GeneratedJob{name='cupid'}
    GeneratedJob{name='donner'}
    GeneratedJob{name='blitzen'}
Finished: SUCCESS
```

So this created all the jobs –
Let's look at the main page.

@jeffreymckenzie

The screenshot shows the Jenkins interface with the title 'All Reindeer [Jenkins]'. The left sidebar includes links for New Item, People, Build History, Edit View, Delete View, Manage Jenkins, My Views, Credentials, and New View. A 'Build Queue' section indicates 'No builds in the queue.' Below it is a 'Build Executor Status' section showing '1 Idle' and '2 Idle'. The main content area is titled 'All Reindeer' and displays a table of build jobs:

S	W	Name ↓	Last Success	Last Failure	Last Duration
1	all-reindeer-dal	all-reindeer-dal	1 min 29 sec - 1/3	N/A	0.26 sec
2	blitzen	blitzen	N/A	N/A	N/A
3	comet	comet	N/A	N/A	N/A
4	cupid	cupid	N/A	N/A	N/A
5	dancer	dancer	N/A	N/A	N/A
6	dasher	dasher	N/A	N/A	N/A
7	donner	donner	N/A	N/A	N/A
8	prancer	prancer	N/A	N/A	N/A

Icons for each job name are shown to the left of the job names. A legend at the bottom right provides RSS feed links for all, failures, and just latest builds.

There they are.

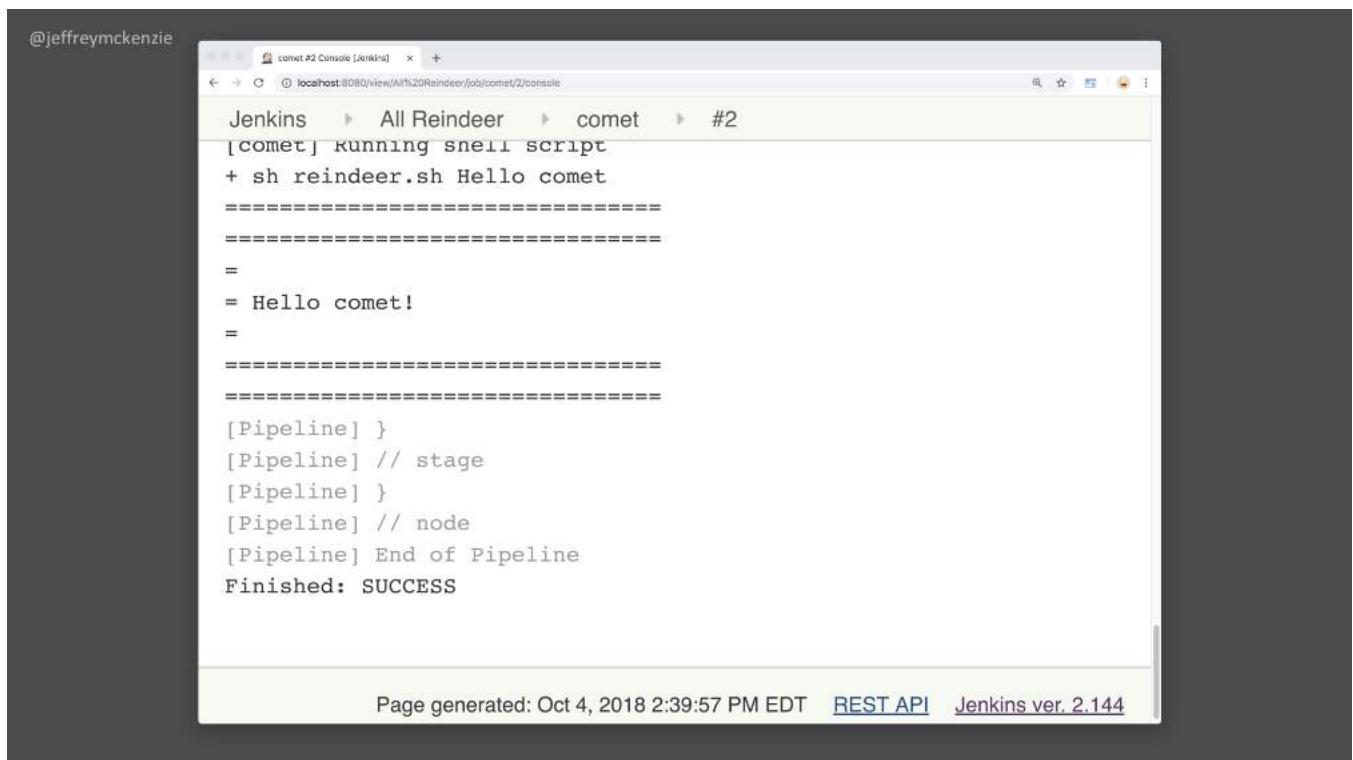
Now we can drill down into any of these
And run them, and they should
Function just as if we created them manually...

@jeffreymckenzie

The screenshot shows a Jenkins pipeline project titled "comet". The URL is `localhost:8080/view/All%20Reindeer/job/comet/`. The page header includes "Jenkins", "All Reindeer", "comet", and "ENABLE AUTO REFRESH". Below the header, the title "Pipeline comet" is displayed, followed by the subtext "Say hello to comet.". There are two buttons: "edit description" (with a pencil icon) and "Disable Project". A "Recent Changes" link is accompanied by a notepad icon. The "Stage View" section contains a message: "No data available. This Pipeline has not yet run."

So let's look at comet –
Hasn't run yet...

@jeffreymckenzie



The screenshot shows a Jenkins job console window titled "comet #2 Console [Jenkins]". The URL in the address bar is "localhost:8080/view/All%20Reindeer/job/comet/2/console". The console output is as follows:

```
Jenkins > All Reindeer > comet > #2
[comet] Running shell script
+ sh reindeer.sh Hello comet
=====
=====
=
= Hello comet!
=
=====
=====
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

At the bottom of the window, there is footer text: "Page generated: Oct 4, 2018 2:39:57 PM EDT REST API Jenkins ver. 2.144".

we run it, and we get the expected result.

@jeffreymckenzie



So a quick recap of really how little effort it takes
[what santa has been able to do]
To get to this end state...

@jeffreymckenzie

The screenshot shows a Jenkins dashboard titled "All Reindeer". The left sidebar contains links for "New Item", "People", "Build History", "Edit View", "Delete View", "Manage Jenkins", "My Views", "Credentials", and "New View". Below these are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). The main content area is titled "All Reindeer" and displays a table of build jobs. The columns are: S (Status), W (Workload), Name (sorted by name), Last Success, Last Failure, and Last Duration. The table lists the following jobs:

S	W	Name	Last Success	Last Failure	Last Duration
●	●	all-reindeer-dal	1 min 29 sec - #3	N/A	0.26 sec
●	●	blitzen	N/A	N/A	N/A
●	●	comet	N/A	N/A	N/A
●	●	cupid	N/A	N/A	N/A
●	●	dancer	N/A	N/A	N/A
●	●	dasher	N/A	N/A	N/A
●	●	donner	N/A	N/A	N/A
●	●	prancer	N/A	N/A	N/A

Below the table is a legend with three RSS feed icons: "RSS for all", "RSS for failures", and "RSS for just latest builds". At the bottom of the page, it says "Page generated: Oct 4, 2018 2:31:08 PM EDT" and includes links for "REST API" and "Jenkins ver. 2.144".

...of having all these jobs generated.

@jeffreymckenzie

```
- reindeer.groovy

#!/usr/bin/env groovy
def call(Map<String, Object> options) {
    try {
        node {
            String greeting = options."greeting".toString()
            String reindeer = options."reindeer".toString()
            stage("Say Hi to ${reindeer}") {
                checkout scm: [
                    $class: 'GitSCM',
                    branches: [[name: '*/master']],
                    userRemoteConfigs: [[url:
                        'file:///Users/jmckenzie/projects/santa']]]
                sh "sh reindeer.sh ${greeting} ${reindeer}"
            }
        }
    catch (Throwable err) {
        throw err
    }
}
```

We have our pipeline code –

One instance –

Shared across all of Jenkins,

Versioned, In source control

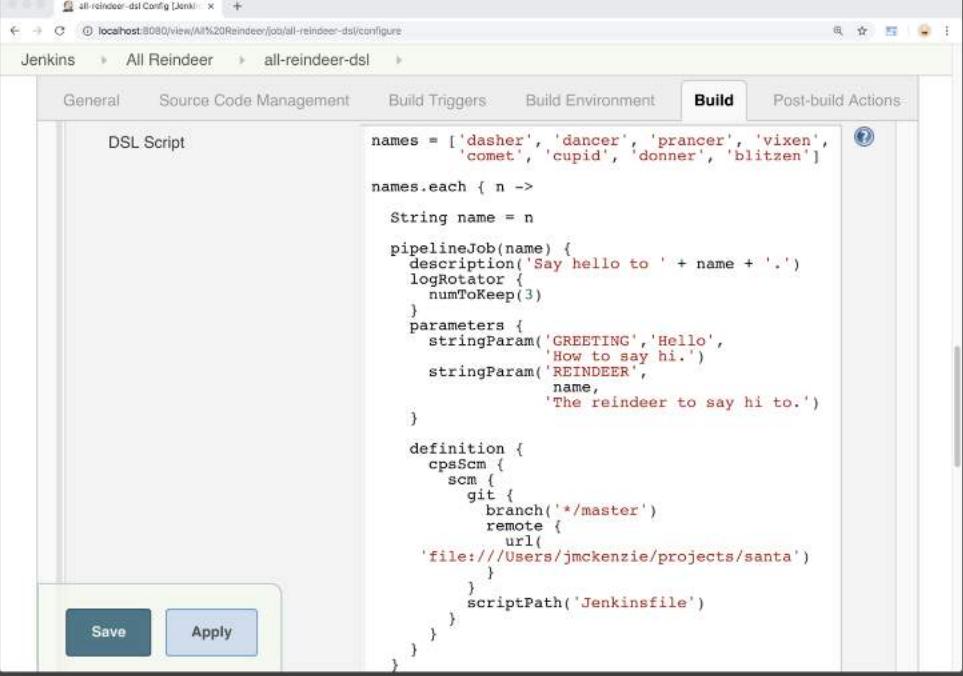
@jeffreymckenzie

```
- Jenkinsfile
#!groovy
@Library('jenkins-lib') _

options = [
    greeting: GREETING,
    reindeer: REINDEER
]
reindeer(options)
```

We have our Jenkinsfile,
Again, once instance of it,
Versioned, and in source control

@jeffreymckenzie



The screenshot shows the Jenkins job configuration page for 'all-reindeer-dsl'. The 'Build' tab is selected. In the 'DSL Script' section, the following Groovy code is displayed:

```
names = ['dasher', 'dancer', 'prancer', 'vixen',  
        'comet', 'cupid', 'donner', 'blitzen']  
  
names.each { n ->  
    String name = n  
  
    PipelineJob(name) {  
        description('Say hello to ' + name + '.')  
        logRotator {  
            numToKeep(3)  
        }  
        parameters {  
            stringParam('GREETING', 'Hello',  
                       'How to say hi.')  
            stringParam('REINDEER',  
                        name,  
                        'The reindeer to say hi to.')  
        }  
        definition {  
            cpsScm {  
                scm {  
                    git {  
                        branch('*/*master')  
                        remote {  
                            url('file:///Users/jmckenzie/projects/santa')  
                        }  
                        scriptPath('Jenkinsfile')  
                    }  
                }  
            }  
        }  
    }  
}
```

At the bottom left of the script editor, there are two buttons: 'Save' and 'Apply'.

And finally, our one DSL job and script
So to scale this out further
You only really need to add an entry in that array.

So all of that in about the same amount
Of code that you probably throw away each day.



Automating Your Automation The Care and Feeding of Jenkins

Jeff McKenzie • jeff.mckenzie@insight.com • @jeffreymckenzie

??????