Claude Code + Onyx = RAG For EVERYONE (5 Levels)

46 MIN · YOUTUBE · HTTPS://YOUTU.BE/ZL8SFRP1FWM?SI=EWQUFZRP5JSBK1KG
https://youtu.be/zl8SfRp1FWM?si=ewqUFZrp5jsBK1kG

SUMMARY

The video discusses the use of Retrieval-Augmented Generation (RAG) systems, emphasizing that many businesses may not need them. It explains when to use RAG, how it functions, and introduces an open-source solution called Onyx for implementing RAG effectively. The presenter outlines a tiered approach to determine the necessity of RAG, focusing on data organization, retrieval accuracy, and the importance of testing for reliable outputs.

- *Many businesses may not require RAG; simpler solutions might suffice.*
- *RAG is beneficial when dealing with large, scattered datasets that need accurate retrieval.*
- *Semantic search helps find meaning rather than exact keyword matches, but can struggle with accuracy.*
- *A hybrid approach combining semantic and keyword search improves retrieval accuracy.*
- *RAG is necessary when data changes frequently, requires verifiable citations, or when building products for end-users.*
- *The tiered approach involves starting with basic chat functionalities and escalating to more complex systems like Onyx.*
- *Onyx is an open-source solution that allows for customizable RAG implementations.*
- *Continuous testing and monitoring are crucial to maintain accuracy as data evolves over time.*

Everybody's telling you to use Rag and I think that's an entirely wrong approach because you might not even need it in the first place. In this video, I'm going to show you why you use Rag, when to use it, how it works, and I'll walk you through a really simple open source solution that you can use with Claude. So, by the end of this video, you know exactly what you need to do and whether you need to do it. So, like I said, most businesses don't actually need Rag at all. I think it's just been put in your head because YouTube. So, what we're going to do here is we're going to go through each layer of solving a problem of chatting to your data because that's essentially what people want with Rag at a higher level when they think about it.

They want to be able to have a conversation with Claude and know accurately that whatever they're getting back from their own internal data is what they need. So, we need to start this video with a mental model shift because most models have massive context windows now. Meaning, you could put in an entire book and still get accurate queries from it. We'll get into when this starts to fade somewhere down the line, but for now, just understand that using this long context window in your chat with Claude or Gemini, whatever it is that you're using, the goal here would be for a deep understanding of one complete source. Rag only really comes into play when you have lots of documents scattered everywhere and you need to have a better job at repeatedly finding the right evidence from a large or changing pile of information. And you need to remember this throughout the video because it pretty much summarizes the entire point of when you would want to use rag. That auditable

approach and the repetition behind that form the essential parts of when we would want to switch to rag. But before we can get into rag as a whole, we need to understand how it works at a high level under the hood. So with traditional databases you might use keywords to find specific results. So for instance name equals mansel we can easily go and find that thing and it is super accurate. The problem that arises though is say for instance we wanted to search through a bunch of logs and we were looking for checkout broken but in the logs we didn't have a table or a database row or anything like that that showed checkout is broken. Instead we had a log file that tells us the payment processing failed at the cart stage. If a user had to go and query this and say check out broken using keyword search, we would never be able to find out anything inside there because there is no match.

The words do not overlap in any way. That's where something like semantic search comes into play because it looks for the meaning behind it. So if a user has to type in checkout broken, if we look in the log file where payment processing failed at card stage, there's a match over there because these things mean the same thing or more importantly, they live in the same meaning space. And we'll get into that in just a second.

Then we go one layer deeper over here and you can see we've got our example for the payment processing failed. And this would go into something called an embedding model which is really just a small AI that then turns this chunk of text into these numbers over here or vectors. And you can think of these as coordinates in this meaning space that we're about to get into. The point is over here we have turned this text into this which helps us understand where it is going to sit inside that meaning space. So these numbers over here they represent what the chunk means not the words that it uses. And that brings us into this meaning space over here. And you can see everything is kind of grouped together where it means the same. So for checkout issues, we might have checkout broken, payment failed, cart timeout. They all occupy this little meaning space over here because that's where their mathematical coordinates are located inside this vector database. Same thing for on boarding. They might live on this side over here. And then billing might live down here. So essentially all we are doing is mapping meaning to coordinates.

And if you look over here with these little dots, the closer these things are together, the more similar they are in meaning. So these dots are obviously far away from one another. And that's how the AI knows when it goes and searches in here that if it was looking for something to do with checkout issues, it's not going to come down to this meaning space over here because it knows they are not related. Something important that's to know about this though is that it's not a pass or a fail type of thing. What it does is it's actually ranking the similarity. So if you remember these little dots from this slide where these things are closer together that is because of ranking. So for instance if we had let's just say these first four rows over here these were all renewal contracts. You can see that they have a similarity score that are closer together. Whereas the things where we're talking about the employee handbook or the office holiday calendar their score is very different. So they get pushed down to the bottom of the list. Now these things are all configurable. Generally a lot of models out there they do a top 5K. So they go and search through the top five things.

You don't really need to know all of that right now. You just need to know how this ranking actually works because it leads us into understanding the problems that can come from just using rag by itself. But semantic

search can actually struggle because of that ranking that we just spoke about. So let's say we have our user over here and they're trying to find a contract from Acme Industries about the renewal terms. Now of course we can semantically search for that because all of the things with renewal terms would occupy that same space because of their similarity score.

the accuracy issue comes into play because he's specifically looking for Acme Industries. That could be living further down in our rankings list. And if our top K is set to five and this thing's actually in six, it wouldn't find this. So, it's less accurate from that perspective. It's not impossible that it wouldn't do it accurately. But what we need with this type of system is the accuracy. That's why we're bothering with RAD. We want to know that exactly what we're searching for is accurate.

So, we fix this problem by taking a hybrid solution. And that's where hybrid search steps in because we can mix the semantic search or the meaning where we're querying renewal terms that matches the contracts, the renewals and the agreements across multiple documents with the specific keyword search where the query would identify acne industries specifically. So even if this thing was ranked a little bit lower, it would still be able to search through all the relevant documents, but then it would be able to match it perfectly because it would look for the keyword of Acme industry. So, by combining both of these methods, it returns the right Acme contract, combining meaning and the exact keyword match. And really, if you're looking at most modern RAG systems out there that need this kind of accuracy, it's going to be taking a hybrid approach for this specific reason. And of course, there are a ton of other things that go into rag, this is just a highle overview so that you can understand some of the concepts that we're going to be talking about in just a little bit. First things first though, we need to understand when we actually need rag because like I said in the beginning of this video, most businesses absolutely don't need it. there are far simpler solutions and that's why we're going to get into this tiered approach.

But you can use this as a compass to help drive you there before we get to those steps. So the first thing here is if the corpus is too large for a single context window and you need to use it over and over again, that's probably a definite sign that you need to start looking at rag specifically because of that accuracy issue over time. With one request, it's really easy to just have a chat with Claude, burn 500,000 tokens, and you would find your answer. It's not entirely accurate, but it definitely works for oneshot type of answers. The problem happens when you try and get back multiple answers within a single question, even if it's still inside the context window, that's where accuracy starts to dip a little bit. More importantly, if you start to have multiple conversations within this context window, that's going to be 500k a pop, if not more, each time you traverse to the conversation, which means your accuracy is going to drop each time you have that conversation alongside other problems. And this brings us on to signal number two, which is having verifiable citations. A lot of people who have these systems where either a customer is talking to them or even an internal chatbot, you need to be able to say that this information I'm presenting to the user is 100% accurate.

And part of having that is having not just this order trail, but also a little source toggle where someone can see what document this information got pulled out so that they can verify that it's accurate as well or perhaps even read the entire document if they wanted to. So, especially if you've got two of these things stacking together,

you know that you're definitely in rag territory. Then point number three, if your data changes frequently, that would be another signal that you might want to start looking at rag. Again, I wouldn't say that point number three would be the one that would start all of this for me, but certainly if it's tying in with point number one and point number two, it's just adding more favor towards using rag because rag caters for indexing new data as it needs to within the system as opposed to manually uploading this thing to your conversation window every single time you need to have a conversation. And then finally, if you are building all of your information into a specific product for someone out there, like an end user, then you would definitely start to look at rag because you need all of these things. Your data would be changing frequently with updates. You would want verifiable citations because the people using your product would obviously need to know that they're getting accurate information and it would certainly be too much for a single context window.

So, when you mix these things together, you can see how you would definitely need rag. More importantly, if you have these first two as a need and the examples that we're about to cover don't cut it for you, that's when you would start looking at it. And that brings us into level one. Now, I always prefer in business to take a constraint level approach. Meaning that I'm only going to go up layers as soon as I run into a problem or a need. I'm not just going to go and get rag because someone on YouTube told me to get rag. I'm going to look at how I work every day or what my needs are and then say, "hm, this is no longer working. I need to go to level two." So, we're starting with level one over here, and that's just using Claude in the chat window. You can easily upload your little PDF inside your chat window and ask it questions on that.

Context window is perfectly fine at doing that. problem only starts if you need multiple facts from this over a longer conversation because then the accuracy is going to start to degrade. But for most PDFs out there that are like 30 pages or whatever it is that you need to analyze. Using Claude in the chat is perfectly fine. You're honestly not going to run into any problems. In terms of the cost over here, it's obviously included inside your subscription. So you don't have to worry about that. And you want to start looking at level two when you want to search the same PDFs over and over again for specific things or just have it as a knowledge base that the agent can query when it needs to. That's when I'm going to start looking at building something more defined. Maybe something like a project. And that's partly why Claude built projects for us is so that we can have this workspace with all of our knowledge and all of our tools and the things that are specifically related to that lane that we're currently working in. So if we look at what Claude projects actually provide to us, they give us an individual workspace where we can have our context and our knowledge and any references that this agent would need in order to go and do its thing.

And I think when a lot of people are looking at a ragbased solution, that's all they're looking for. They're looking for a way to chat to an agent in a meaningful way that actually has the knowledge that they need in order to complete a task. And Project is perfectly fine for that and actually a form of rag in itself because after you've hit your 1 million context limit over here, you actually get 10 times more than that because of how the backend works within projects inside Enthropic. It's a little bit of a black box in exactly how it works, but it is raglike in the nature of what it's doing, which allows you to have these longer conversations with it based on the specific knowledge in here that you add, whether it's PDFs or markdown files, whatever it is. You also have the ability to add instructions to this, so you can tell it exactly what you need in a prompt over here. And if you follow this channel for a while, you've seen we've been turning projects themselves into an AI operating system using

co-work. If you haven't seen that, it's in the description below. And that's where you take it to the next level from just being this static place where you'd have a conversation to ask about your knowledge and turn it into a system that actually builds things out for you. But I don't want to veer too far off course from this actual video. So let's come back to this slide over here. You are over here working inside your project.

This is still part of your subscription. It's not going to cost you anything more as long as you're using it inside here. If you ever did need to share any of the stuff that you're working on here with a teammate, there's a share button inside the project that you can easily send to anybody on your team plan. You can also export anything you put in there into Markdown if you did want to share it. Of course, it's not the most efficient thing to do, but it doesn't mean that you should go and get rag. So, when would we actually move up to level three? And I think for me, the biggest takeaway here is that if you need configurable retrieval, as in you want to have sourced citations and you want to be able to configure how this thing is working under the hood, of course, Claude projects are not going to cut it for you. So, that would give you one tick box to moving up. If you needed autograde citations, that's going to be your second signal. And definitely if you want live autosync from many systems, you would start to look at that. Now, I realize that projects can totally use MCP to connect to most software out there where your data and knowledge lives. But again, if we're talking about large amounts of data that spanned years, that's not going to cut it. It's one going to eat through all of your tokens, but two, it's going to be massively inaccurate. So that means we need to start looking at something more reliable at gathering these larger amounts of information and codating them into something that you can use. And that brings us on to level three, which is Notebook LM. And it is a really good solution for most people who again want this chat functionality without the complexity of setting up rag because again this is using a form of rag on the back end. It's not entirely configurable at all but it does give most users what they need slightly different to projects. Obviously it has a few features that projects doesn't and it serves a different use case. But in terms of chatting with your actual data it is very good at it and more importantly it gives us citations here.

If we flip on over to my notebook, it's very easy to set up. It's included with Google and it's entirely free. There are obviously paid tiers, but even on the free tier, you can get away with quite a lot, probably more than what most people would actually need for their day-to-day use. The point is though is that you can put in totally different forms of The Point [clears throat] is though, you can put in many different forms of data and you can also hook this up to YouTube. I have videos on how this can be useful if you're building apps or trying to connect to separate systems. They'll be linked down below as well. More importantly here once we've put in our coming back to the rag use case though once we've put in coming back to the rag use case though once we've put our sources in we can just come down to this little box over here and question our data anywhere that we want. So I've got some information in here on habit gathering. So, what's the best way to get habits nailed down?

And then it will go and search through all of this. And as a part of doing that, it's going to give us the citations that we actually need so that we can go and check out specific parts of the video or read specific parts of the PDF that we would need to. So, if having this audit trail is very valuable to you, this is going to be one way to do it. But more importantly, I definitely think that at the time of me recording this, it's better than projects specifically for the users who want to be able to speak to their data in a meaningful way at an accurate level. And so you can see here it's answered my questions. And then for each of those questions, it's givingven me citations of where it got this information from. So again, that's really valuable. And then you can do various other things

with your data.

We're not going to go into that in this video. So like I said, notebook is probably going to be fine for the majority of people out there. But at this point, we're probably going to have to start looking at rag as we get to level four because we've now identified that we need more control and we have tons of vast data that we need to have an order trail for amongst a few other things. One of which is because notebookm does not allow us to configure the accuracy across those multiple systems and also at the time of me recording this it does not allow API access with an outside AI. There is an easy way to get Claude hooked up to it.

I have a video on that as well, but it is outside the terms of service of what Google actually allows. So I would never recommend that to a business. It's perfectly fine if you're flying solo and you're willing to accept the risk. I just created a dumb account and that's how I get Claude to work with Notebook LM. But now we need to get on to rag and that brings us on to level four. The solution I'm putting forward is Onyx. It is entirely open source. There is a paid for version as well. And I think for most users out there who are nontechnical and don't need this massive production grade thing, Onyx is going to be the solution for you. This is what we're going to be setting up in this video. But but before we get into it, I just want to talk a little bit about it because there are things you need to know. So when we're at this level, you're definitely going to have a team.

It's no longer just one person who needs this random solution. You have found that you have a lot of data. It needs to be searched in an accurate way across multiple systems with an evidence-based approach. The cost of this thing depends on the path that you take. So, it is open source. If you have a spare server lying around, like I've just set this thing up on my MacBook Pro, runs perfectly fine. Obviously, the more data you have, the more scale you need, it's probably not going to cut it in the end, especially if you're using your laptop for other things as well. So, you would definitely want a dedicated server, whether that's a VPS or something that you've got hosted on actual hardware in your house. That's a perfectly reasonable approach. The other approach is to use their cloud options. They obviously offer this as a service. It's about $20 a month per user, but then they take care of all of that. So, if you're looking at this as a business, if you don't have the technical capability or you don't even want to bother doing that, taking the cloud route makes most sense because I think $20 is pretty reasonable and it certainly removes you from having to deal with that headache yourself so that you can just get this thing done. But if you're not that way inclined and you want to set this thing up yourself in your own environment, that's what we're going to be covering in this video. Before we get into there, we need to take a look at level five.

And I think if you are here, this is where you would actually get somebody who knows what they're talking about to come and help you implement this because it's probably going to be a much larger solution that you need to cater for. This will be things like Pine Cone, Superbase, Quadrant. Of course, you can use these in the same way that we're just about to use Onyx, but for me, I chose Onyx because it's much more userfriendly for the average user who just wants some kind of plug-and-play simpler rag. These solutions over here are much more configurable. They're definitely power user oriented, even if you're using Claude to help you get things done. And when you're at this stage, you're probably looking at product engineering as opposed to just a solo founder trying to run their AI operating system and have this accurate knowledge base behind you. And so here

we are, everybody's favorite part of the video where we get to the tool porn.

Now, there is still going to be some theory while we're walking through the playbook because obviously I want you to understand things around testing and why we're doing what we're doing. Setting this thing up is super easy. But first, we need to take a look at why I chose this product for most of the people out there. They're claiming to give your team superpowers and that's because this is more than just for rag. It has an entire chat system that we can use not just to chat to your data but also just to chat to an LLM in general. But more importantly for those of you where data privacy is very important. You don't have to use claw or codex. You can use your own local model because this thing is entirely model agnostic. If you wanted to, you could also give them the capability of web search. You can also crawl websites using firecrawl and other MCP plugins. And it also has deep research functionality which is super important when you're trying to gather more of that knowledge to fill your database. There are also a ton of other features inside this thing, but we'll get that as we uncover each layer of using this program. So, in terms of getting it, if you want to pay for the cloud version, obviously you come over here and look at pricing. They have a business tier with a free trial, so you could see if this thing actually works for you and then enterprise, you're probably not watching this if you're from an enterprise. Point is, if you don't have the technical capability or you don't want to deal with any of that stuff, paying for this is a no-brainer.

$20 a month per user, I think, is perfectly reasonable and it unlocks everything that we are about to set up manually. Now, if you do want to go down the self-hosted route, don't be turned off because it is extremely easy to do with Claude. I didn't have to do anything manually. All I did was I clicked on over here, which is their GitHub repo, and this takes you through to this screen where you can grab everything entirely for free. This is under MIT licensing, so you need to take that into consideration, but really, you can get away with most features for whatever it is that you're trying to run. We'll have a look at what you don't get towards the end of this video, but it's honestly nothing that most of you guys are going to be using. In terms of getting claw to install it, all I have to do is copy this URL. Then I head over to my VS Code environment. You can see I had a really long chat with it while I was setting all of this up. If we close these other messy windows, I would just come here and say to it, I want to set up Onyx on my local drive. Can you please check out this GitHub repo and install everything for me? Now, what it's going to do is it's going to tell you about several dependencies. I'm not going to rerun this cuz I've already done it. I would hit enter and it would tell me, okay, I've read through the readme and what we need to do is make sure that we have Docker. So, you're going to need to get that. And when I say you, Claude is going to get that for you. Once it's done that, it's going to understand all of the requirements of this GitHub repo. So, it will know exactly how much RAM you need as a minimum requirement, what the recommended requirements are, and based on whatever system you're installing this on, it will do an analysis on that.

So, for instance, mine is on my MacBook. And this thing said to me, hey, you've limited your Docker to only having 12 gigs of RAM allowed. You need 14 for Onyx to function correctly. And it was at that point that I then told this thing to just do whatever it needs to do in order to get the job done. Of course, if you're in a real business and not a dev environment, you would want to go through this accurately and make sure you know what you're doing, but it's really just logical things as opposed to rag based technical implementation at this point. Eg. Make sure you understand where you are installing this, why you're installing it there. Like I said,

have a dedicated VPS or dedicated server if you have that capability so that this thing is always on and people in your business can search. I'll have a guide in the video below, so don't worry about having to remember every single thing that I say in here. See, here we go.

Guide. I told you I had one for you. So, we're in the install phase at the moment and this is where we're going to install it on Docker like I just mentioned. There are two types of Onyx that you need to understand. One is the light version. Don't install that because that's literally just the chat interface without all the rag stuff. And we're obviously here in this video for rag. Then the installer is going to run through the stuff that we already spoke about. And realistically, in less than 4 and 1/2 minutes, I had everything set up in the way that I needed it to. And here you go. You can see Onyx is up at my local host. All 12 containers healthy.

So that's inside Docker if you don't know what a container is. Point is, after it did that, it then told me exactly what I need to do in order to get the first part of this ball rolling and that is to set up an LLM provider. That's really important because again, remember this thing doesn't ship with an LLM. You need to put your own in there. So, you will need to pay for an API either from Anthropic or Codeex or your local model, whatever it is that you're using. You need to give it that. So, we just click on here and it opens our Onyx deployment. You can see that we are now met with the shiny screen that looks familiar to anyone who's ever used an AI web gooey. For me, I've already added Claude to this via the API. You would add whatever you want. There'd be a little box here that says add your AI now. Otherwise, we could head on over to the admin panel, which is what we would want to do because we are the admins when we're setting this thing up ourselves. If we come on over to language models, you can see the default model is Opus. If you had multiple models, you could obviously change that.

All of your settings for anything that is configured here. So, if you only wanted to present users with specific models, you can also limit it that way. That's really good for people who like to use Opus for everything. you wouldn't want them burning through all of your cred. You could limit them to using Sonnet to do specific tasks, whatever it is that you need at the time. Then, if you wanted to add more providers, you could do that over here. And of course, custom models and things like that. I'm not going to focus on the other things in here with this video. We're particularly focusing on rag. So, these other shiny features we'll cover in another video. But for now, what I want you to do because we want Clawude to do most of the work within our environment is we want to give this thing API access. So, we need to come on down to where you see integrations over here.

You can have a service account and this is exactly where I've set up Claude. So you would just create a new service account, put a name and then all you would need to do is give it account permissions. In this case, we want to give it admin credentials so that this thing can access everything via the API and do what we needed to do, especially if you're a non-technical. Once you hit save and you've put your API key in there, you will have Claude connected to this and able to do whatever you would be able to do via the admin console.

Something to note here that's really cool is that you can connect this with Slack and Discord. So if your users wanted to use the rag feature in here, they wouldn't even need to use Onyx in the sense of logging into it, they could just chat to it directly from within Slack amongst a few other really cool things that this thing can do when you give it Slack integration. But for now, we're going to look at the next step inside our playbook. Okay, so

now we're in the pre-prep phase because you're not just going to go in there and suddenly throw everything into rag and expect that you get this massively amazing product. What we need to do is some pre-ingestion data prep. This is one of the most important things because whatever you put in is what you're going to get out. So if you put junk in it, you're going to get junk answers out of this thing. So we need to focus on making sure that our data is not just fresh, but also that it is accurate from the get- go. Otherwise, we're going to end up with trash. So it all starts with a data inventory or mapping your data.

You need to understand where your data lives. So every single system, is it in Drive? Is it in Notion? Do you have Obsidian or a team share? Whatever. All of the pieces of the information that you want to gather, you need to map that stuff out. And of course, you can get Claw to help you with this. I'll cover this in a separate video, but the TLDDR is that you can just connect all of your APIs or MCP to Claude and then you can get it to traverse those systems for you and pull out what data is living where, put it all in a spreadsheet or an air table, whatever it is that you're using.

And that way you can start to form your data map or inventory. You want to make sure that you're not guessing here because that's obviously going to set you up for failure. Then what you want to do is you want to make sure that as part of this inventory, Claude walks through every single one of those folders, but it organizes it by type, by size, and by quality because Claude can also be the judge of quality before you go and throw this thing into a rag database. And more importantly, you need to have triage system. So you can use three types of buckets. That's one of the way to do it. You can have green, yellow, and red. Green is obviously good to go. Yellow needs a little bit of work, and red is probably something that you just need to either fix or get rid of. You'd be surprised how many organizations have that red bucket and they think it's absolutely valuable and it's a problem that we all have. We think all of the data in our business is absolutely valuable. We can never get rid of it. And mapping this helps uncover where the junk lies so that you don't end up having that problem. As a part of this process, you would also want to make sure that you cater for any duplicates. So you need to run dduplication. You don't put the same information in twice. That would be a waste. Also remove any PII that you don't want in that system. It's very easy to scrub that with AI. There are several tools that will help you do that as well. But more importantly, you might just think to yourself, okay, if I need to scrub this, does it really need to be in there? Maybe I should just leave it out entirely. Remember that in this case, you're using rag to talk to all of this data. So, if there is something in there where you sit down and think, okay, no one's ever going to need to speak to this, it probably means it doesn't belong in there. Something else that you might want to do, it's not entirely necessary, is convert PDF to markdown or something like that. Most systems like Onyx, Pine Cone, whatever, they're all really good at modern day PDFs, so it's not the biggest problem like it was a few years ago. But if you can make things more efficient, that's probably the way to go. And then we would also want to look at things like metadata tagging and PI scanning like I already mentioned. And that brings us into the next part of this where we're actually going to be looking at all of these documents that we've now organized. We're going to be connecting to systems that we need to. And then we're going to be building any document sets that we might need to connect to the agents. So I'm going to head on back to our admin panel over here. And just note that Claude could be doing everything I'm about to be doing on screen right now. I'm just doing it to show you kind of have a visual journey.

If we head on over to the existing connectors, you can see that we've got over here documents and knowledge.

Now you can add your connectors so you can choose whatever it is that you would have your data in. Again, don't give this thing access to everything that that it doesn't need access to. That's just silly and setting you up for failure. So choose the specific things that you want done first. And I would also do this in a scaled approach. I wouldn't connect everything all at once and just go in start with something small. Make sure connectivity is working and actually test this for yourself. See if it's the right solution before you embed your entire business in this thing and just waste your time. Now in my case for this demo, I'm not going to connect any of this stuff. It's just like connecting MCP via anything. You would create a new connection over here via the credentials. You would need to set these up either using OOTH or a service account. You download the JSON file and you upload it below. Now, that's really easy to do because Claude can obviously do that for us. If you're connecting via CLI to pretty much anything else with Claude, that's the process that we are using here. Obviously, depending on the type of connectivity that you'll be doing, they'll have different systems.

So, for Salesforce, you can just use a Salesforce username and password and a token and various methods for all the different systems. This one just needs a token. if you're going through via GitHub. For me, all I did was go with file for this demo process. So, you can call this whatever you want. Let's just call this demo. And then I would drag and drop any files, but I've already done that because it took a while to upload these things on my laptop and I don't want to spend 28 minutes waiting while I film this video. Something to note though, if we're looking at systems like Notion or most of these systems, to be honest, what this thing is doing by default is polling them. So, it looks for fresh information every 30 minutes.

And there are a few things that we need to understand about that when the time comes, but just know that that's a thing. Once I've uploaded my data, in this case I just uploaded a bunch of fake PDFs, and I did that specifically for this test. But once I've uploaded my first chunk of information, what I might want to do is create a document set. And you can see here, this is just a way of logically grouping any of the information that we have in here. So we could create a new one, and we could just call this something like sales information. Give it a description, blah blah blah, add our connectors in here, and then we would have this document set created. So if you wanted to break down specific logical groupings for sales members or content creators, whatever it is that you needed to logically break down, this would be the way that they would only be able to search specific documents. You can of course obviously create separate user accounts and you should and each of those user accounts would have their own level of access and the things that they can do and can't do and stuff like that. But this is a really good way and you should be doing this if you want that separation, but also just something that's grouped logically. Depending on whatever it is that you're running this thing on, it can take quite a bit of time. Like I said, on my MacBook Pro M4, it took about 28 minutes to upload eight PDFs that weren't really that big, but a fair chunky size. It's very CPU heavy, so your computer's going to make a ton of noise. If you're using a VPS, that's pretty chunky. I guess that wouldn't really matter. And obviously, if you're using their cloud-based solution, you're not going to have to worry about any of that. I imagine if you're uploading something ridiculous like years worth of documents, you need to make sure that you have the space for this and that you actually have the patience for this because it's going to take a while. I think the size is about two and a half times larger than the raw document size.

So you would need to factor that in if you were hosting this yourself and really dive into the requirements for your data size and what you need to cater for with the hardware side of things. That would also mean

understanding your retention requirements. So how long are you going to keep this data? How much is it going to grow each day that new information comes in over a period of time? Then in terms of index settings, realistically most of the nontechnical folk are never going to touch this kind of thing because it probably wouldn't make sense to them. That's kind of where they've catered for this to make this somewhat userfriendly. For me, straight out of here, I changed absolutely nothing. And every single test that I ran it through, I got back exactly what I wanted in the most accurate way. Now, of course, depending on the vast amount of data that you might be adding to this thing, you definitely want to be testing it.

You definitely want to be monitoring it, and we're going to get into that in just a second. But for the majority of you, you're probably not going to be touching these settings. if you do or you did want to understand if it made sense for your specific use case. Remember, Claude not only understands the exact documentation from the repo, but it also understands rag in general and it can talk to Onyx directly because we've given it API functionality. So, if you put all three of those things together and you give it the context of your own business or the thing that you're trying to do, it can help architect a solution or change any of these settings if you need to. One of which might be this contextual retrieval if you start running into failures. This setting over here essentially makes Claude far more accurate. I think it was about 35% based on what I was reading online, but it is obviously going to come at significant cost increase because this thing is adding document level context to every index chunk so that it improves that accuracy. So there is obviously a trade-off with that and again that's where you would select your model to take care of this. Like I said though most of you probably won't need to use this. Then like I said earlier there is also image processing. So if this was really important to you, this thing can extract embedded images from uploaded PDFs and all that and then it will create a summary using a vision capable model. In this case, we all know that Claude is now very good with addressing images. So that could definitely be a viable solution if you needed this sort of thing. You can easily turn it on over there. Okay, so we got our data into Onyx now and obviously we can go and search and do what we need to do. But how do we know that this thing is actually accurate? Because if you're asking it a specific question, remember that these models already know a bunch of stuff about the world out there. So even without web access, they can generally answer you. So if you had to come in here and ask a specific question that wasn't only related to your data, you would probably get an answer. It might be without a citation, but you would get an answer. And that can trick a lot of people into believing that this thing is actually working. But that's not what we do here. We need to test everything that we've put in here for accuracy. So what we're going to do now is we're going to look at the different ways that rag can actually fail and then we're going to look at running some eval so that we can address those and make sure that we're actually getting accurate responses. And so we're back at everybody's favorite slide decks. Told you there was going to be more theory.

If we look at failure mode number one, the chunks can split midthought. So if we have our thought over here, this is our [clears throat] example document and we have the deprecation date for endpoint X is March 15th. Now, if this thing is chunking and you haven't set up the chunking correctly, what can happen is this can actually get cut mids sentence. So, you would have chunk A with the deprecation date for endpoint X is and then chunk B would be the actual date. So, this thing is missing the context over here. It's just a random date. The issue that creates is that neither chunk has the full answer. So, retrieval is only going to find half of these things and the model fills in the rest of the wrong answer. And we obviously don't want that because that removes the accuracy from our system which is why we built the whole thing in the first place. Fix is pretty

simple for this. You want to use smart chunking and respect meaning boundaries. So most of the systems that you would be using nowadays, they're actually pretty good at this just straight out of default.

But realistically through your testing and some of the things that we're going to be looking at now, you would figure out if this was a problem for you. And because Claude is connected, it's much smarter than we are. It will be able to understand where you're going wrong based on what you're putting in there. It also obviously has full access to the logs and the API. so it can monitor what it needs to and then adjust accordingly.

One of the most important things here to remember is to always have a little bit of an overlap when you're setting this up. Realistically, you're probably not going to be configuring any of this. You will get Claude to help you if you run into a problem. More importantly, it can do an analysis and tell you if the padding is set up, if there is enough padding, things like that. Second failure here is that chunks can lose context. So, like we spoke about in this slide over here, if it doesn't have specific context for whatever reason, maybe not even because of that problem, you wouldn't have what you need in order to give an accurate answer. So that's where that contextual chunking comes into play. That setting that I showed you in Onyx that I said costs a lot of money if you turn it on. Essentially that solves this problem because what it is doing is giving this little chunk some more context about where it came from or what document it was in. You can see here we have revenue grew 3% but we've added that context at the beginning. So it would do that for every single chunk and that way you wouldn't run into this problem and therefore get this increased accuracy. Failure number three is pretty easy to solve. As we've already spoken about, semantic search misses exact matches. That's why we take this hybrid approach and that's pretty much exactly what Onyx is doing for us already without us having to do anything. Then we get on to the last one here and that is where the AI gives a confidently wrong answer. This happens all the time even with AI in general where it is adamant that you are wrong and it is right and here is the reason why it's right and blah blah blah. That can happen with rag 2 and it's very bad if that happens even once because that means the entire system cannot be trusted and again because we want that accuracy we need to make sure that we can trust the system. So we need to game this thing right and we do a bunch of tests to do that to find out if this thing would stop if it doesn't know an answer and say I don't know that or if it's going to do whatever it can to try and please you and just give you an answer and then make up a whole bunch of trash. So at this point in my life of course I didn't do anything manually. I didn't sit down there and write my own test questions. That would be ridiculous. What I did is once everything was done and uploaded, I got Claude to analyze the data that was in there and figure out what would be the best questions that we could actually ask this thing. So, it went and did it.

But it didn't just do that. It also explained to me why we're doing it in that specific way so that we can address some of the failures that we just spoke about. So, if we flip on over to the guide that this little thing put together, we can see here it made me 12 questions across four tiers. Each tier tests one specific rag metric. And of course, this would be a lot larger depending on the type of data that you're using. This is just for demo purposes to illustrate a point to you.

Tier one questions address the retrieval hit rate. So what are we testing? Did the right chunk land in the top K results sent to the LLM. If you remember that's the whole ranking thing to make sure that we're actually getting the relevant things sent back to us. We're testing this because if retrieval misses, no model can recover the silent

killer of every rag system. Then our tier 2 questions offer crossdocus. So you wouldn't just have one document and then you have a bunch of them and that's where it's pulling all of this information from from that semantic meaning space. So what are we checking here? Did retrieval pull all of the evidence, not just the most obvious chunk. And we do this because multidoc questions need multi-doc chunks. Partial recall means that we're only going to get partial answers and that defeats the point of this system. Tier three over here is where we are doing dock only knowledge. So we're testing that whole faithfulness thing. Is it going to make up stuff to please us or is it only going to show us things that actually exist in the documents? And we're testing this here because we want to make sure that it is using real citations. We want to be able to click on them and see them and then review that this is actually legit as opposed to the abstension rate, which is where we're testing if this thing gives you confidently wrong answers, thinking that it's right just to please you. Now, on screen, I can obviously do this manually. I'll show you where I typed in my test questions and things like that.

But realistically, if you wanted to test this at scale, you could get Claude to send this via the API, via send message, and actually do some of this for you and then just tell you how the answers were. So, if we come back to our front end over here, you can see that I've got three previous chats. Now, the reason why I have these chats in here is because it was partly due to testing. This over here where I'm asking a specific question about Splunk management. Splunk is just a product.

You don't really need to know about that. But, it's obviously globally recognized, especially trained on by AI. So, it answered this blatantly on its training knowledge. And then I asked it, did you retrieve this from rag? No, I didn't use any tools for retrieval. I answered directly from my built-in knowledge. So, obviously, what we're doing here is you can just ask this thing where it's getting its information from. And that will tell you straight out of the bat. But that doesn't mean that it's entirely accurate. So, after I prodded it to only use its internal stuff, then it went and did it and it told me exactly where it got it from.

These were the two manuals that I uploaded into here. But, it works differently when you have your own internal knowledge. And that's where I got AI to create this fake files for me around this Acme bank so that we could actually run through the test process over here. And this is where I started asking it very specific questions to address the four things that we just spoke about in the previous slide deck. And Claude set up all of this for me. So you could either do this manually or you could get Claude to run these for you and just judge its own output, which would obviously be faster. I would say if you're going to go down that route, you still need to be the human in the loop. need to make sure that Claude is not being inaccurate on Claude's inaccuracy. If that makes sense. I want you to run through our list of eval questions based on the dummy data that we set up. Send it automatically to Onyx. Run the testing, analyze it, make sure that whatever it's done for us is accurate. The audience is watching right now. So, it' be good if you could explain what is happening and why we are doing it. And so this thing is just going to connect via the API to Onyx and send those test messages for us, run through its loop to see what the results are and then we can work with it to make sure that we get the job done properly.

This is really useful for people who don't really understand what's going on here rather than having to read a manual. You could do this on a small set of data to understand why it is doing this. And of course, like I said, Claude is going to be explaining to us what is going on here. That's why I specifically asked it. The goal of this

whole thing isn't to just get everything done for you. Of course, that's part of it, but it's about learning here, especially if you're going to be managing this stuff yourself. You want to know what's going on. Don't just outsource it to Claude because it can be wrong. So, it's running its test questions. Something to note, if you wanted to, you could connect MCP to Onyx as well. So, if you just wanted to use this to chat via MCP in natural language, you could do that, but you can see I can do that just as well with the API. And Claude is busy talking to you. To the audience, I'm about to send 12 questions in sequence.

After each one, I'll narrate what each tier it tests, what we're looking for, and whether it passed. The whole point is to prove that rag works against measurable criteria, not just vibes, which is what I've been saying this whole video. While that thing is cooking in the background, we can just jump back across here and I'll show you what I meant by citations and the evidence-based approach. So, when I asked one of the test questions that this thing is about to be asking, you'll see here where it was searching. So, it searched only internal documents because it automatically knew it had nothing inside its internal knowledge. It shows us all of the documents that were searched. And then whenever it is stating something, it's giving us citations over here. So this is where it find it from that specific doc. This is where it found it from that one. And you'll see this is very similar to how Notebook LM works, which is why I mentioned that it's a pretty good thing to use if you're just a solo person looking to chat with your data. You're not going to have the same granularity and entirely the same accuracy as this system, but it's pretty damn good for what most people are going to be using it for. More importantly, if we come back to the question itself, so why did Acme Bank alerting delay incident on 2026312 happen and what's Hellbird doing about it? Now, these are all madeup things, but if we look at this question itself, we can see the specific things that it's looking for. We're doing this to test how accurate this thing is at pulling out very specific information. So, we have a very specific bank when there could be multiple files from multiple banks. We're also looking at a very specific date over here. And we're also asking it what's the company Halbat doing about it. So it has these three things to take into context. And for each point where it answers this specific question about what happened on this specific date, we would have our citation. And this is how we verify that this information in here is accurate. In the same sense, this is what Claude is going to be doing for us at scale because it can check against these criteria from what it was testing in here. How did it respond to the question that we gave it? And in its response, was every citation accurate and meaningful to the answer that it gave?

So, Eval is complete. Apparent score 7 out of 12. But let me walk through this with the audience honestly because the script told me one story and the actual content tells another. And you can see here it's gone through each of our questions to find if we could retrieve specific things from this. So, this one we were asking between 50 to 200 gigs a day. That's exactly what we got back in one of the things that we looked at on screen already. If we look at number three over here, we were looking for a very specific project code. That was a pass because it returned that very specific project code exactly and cited the exact right document. Same for a principal engineer rate. That was a pass because it returned the exact dollar value that we need and it reproduced the full rate card table. And you can see in tier 2 that all of our tests passed as well. Everything was labeled correctly and we returned the exact information that we needed. So we had no cut off context where we were missing those chunks because they were separated incorrectly. I also haven't switched on that setting for the contextual awareness per chunk. So, we're doing this straight with the default setting, which is a really good sign. On the faithfulness test, this thing named the exact people from this company. So, it named the CESO that we needed. It didn't make up a name around that sort of thing. And importantly, this thing made an audience note for us.

There is no way that Claude is able to invent Daniel Marsh or the 30-day rule, these two things that it found over here. So, this is proof that the rag is working because it is internal knowledge that only we would have. Then, when we get on down to tier four, the initial testing script said that both of these failed. But when this thing did a second review, it actually turned out that they passed. So it didn't make up any information for this and just try and please us, which is the last thing that we were testing for. So when this thing rescored itself, you can see that we actually got 12 out of 12. And specifically for our demo data over here, this was a perfect test.

So the lesson to take away here is when you're going through this with Claude with your own data, it's going to make unique tests for your data. It's not just going to do exactly what this thing did for me. for the most part in terms of the frameworks, yes, but obviously you would have your own questions and your own workflows that this thing will do for you. The important part is that you just understand your own data because as long as Claude can do all of the heavy lifting for you, you need to be able to know what it was looking for and say, "Oh, yes, that is exactly what was written in that document. Even though Claude is doing the initial sweep, you need to be the final human in the loop." And when you've done that, you can easily just ask Claude, "Make me this unique testing plan." Whatever it is you need, you need to add your judgment onto this. Great. So, we've tested all of our data. Now the next part over here would probably be to extend the functionality. So that's where we get into projects and agent.

Now much like Anthropic has projects, we have projects in here too and they pretty much work in the exact same way. We have our instructions, we have our files that we can add if we wanted to and then we can just chat to it as normal in here and you can split these projects out. Again, if you wanted to see this sort of thing, I have videos below specifically on how I do it inside co-work and claude code itself so that we have the AI operating system model. I think that's probably less useful for the cases that people might be using this type of application for. What I do want to focus on though is agents over here because we can create agents for specific departments that are really knowledge aware. So for instance, if you wanted to create an onboarding agent for your business whenever somebody new joined your company and they wanted to be able to ask questions or you wanted this agent to take care of onboarding related tasks in an accurate way, this would be one of the ways to do it.

Obviously, you've just ingested all of your company's HR documentation and all the policies of how your company works. Anything else this new joiner might need, you could then have this little rag database, which is now known to be accurate because you've just tested everything, have a very specific prompt and address a person in a very specific way and then walk them through the onboarding process. They would then also have this chatbot that they could speak to to find out information. What's the leave policy? Who do I speak to to book time off? Stuff like that. It can all be taken care of with this agent trained on your accurate knowledge. And you can see it's got a lot of the functionality of other agents out there. So we can set our prompt down here. We can give it specific knowledge. And then we can get it to take actions. So if we needed it to search the web, which obviously we wouldn't if it's an internal bot. The point is you have the different options here to choose how you would want this agent to interact with the real world or use tools in the real world, including coding, which is pretty handy. You also have the usual suspects like the default model that this thing is going to run on and the knowledge cutoff date which can be really useful if there is a specific time that you only want this thing to have knowledge up until. So again, most of the stuff would need to be planned up front if we're looking at this particularly just from a rag perspective. Setting this thing up for rag is really really easy especially if you got

clawed. The more important part here is architecting the entire solution from your data but also what is the user journey going to be? You need to group your users. So if we come on back to the admin panel over here, you'll see that some things are closed off to the paid version and perhaps even the enterprise version. So you'd need to factor that in depending on the journey that you're going to be setting up within your business. But you do have obviously the ability to add new users here. You can invite them via email. And once you've done that, you can obviously assign them a role. Most of them will probably not be admins unless they have to do anything inside this panel, but but really they'll probably just be general users. Then depending on whether you're using the paid version or not, you can obviously split these people out by groups. And then you can cascade their permissions in a way that makes most sense for your business so that they only have access to the things that they need access to. Then just to wrap things up here, just because it worked once when you first set it up doesn't mean that it's going to work forever. You need to make sure that you're monitoring this environment. More importantly, that you're actually doing tests after you've already done the tests initially. The reason we do this is because your data can drift. You're going to be adding new data to the system. Things will change over time and we want to make sure that the accuracy that we had in the beginning is still there after all of these changes come in and the new data has come into play because your system is going to get larger and larger and perhaps things will start leaving your system. Whatever kind of retention you have set up forms a part of this picture. So we need to monitor this and one of the easiest ways to do this is to actually just have a weekly eval rerun.

You can set that up with Claude. Say every Monday go and run these same 12 tests that we just ran on specific things obviously and compare it to the week before. These tests will obviously need to test different things depending on all the different types of data that you have in your system. But again, this type of thing can be planned for once you've gotten your data in. If Claude has been a part of your journey from the beginning where you were doing your data mapping, it will have all of that information that it needs in order to create the tests that we ran through with this. Once it's created the test, it will have both forms of context that it needs in order to build a proper monitoring solution for you so that you can have this accuracy over the weeks to come. And that's why it's so important to get Claude involved in this journey.

Obviously, if you have PII and very important things that you don't want Claude to be analyzing, you need to make sure that you're not throwing that stuff into Claude because otherwise you're breaking your whole data privacy thing. But obviously, you can opt out of training anyway inside your Claude subscription menu. Point is, make sure you know what you're putting into AI before you put it in there. And once you got that going, then you can use this thing as your co-pilot to walk you through your entire journey. But I think that pretty much wraps it up for this video. I will go into deep dives specifically into testing and various other things related to this rag system in other videos. I don't want this to trail on forever. So I hope this video was helpful. Leave some comments below if you have any. Otherwise, check out the videos on the screen right now.

They'll definitely help you in your journey. Thanks very much for watching. See you guys in the next one.