L nnec i ns i in i e

HCL Connections 7 Sizing Guide

The Scope

To successfully run HCL Connections 7, one needs to assure that IBM WebSphere Network Deployment is available as well as one of the supported backend databases. This document will give some inputs on how to decide based on performance tests performed by HCL (and IBM in the past) and best practices learned and confirmed by multiple customers through time. However, it is important to understand that a silver bullet, which would apply for everyone, doesn't exist, and that in some cases you need to consider specific components in a different way.

This document does not consider HCL Docs.

Understanding HCL Connections 7 without Component Pack

Understanding the defaults

HCL Connections 7 consists of 21 Java applications which are running on top IBM WebSphere ND.

Each of those applications can be run as a standalone JVM inside WebSphere, or as a part of a single JVM, a so called cluster, on top of WebSphere as well.

Each JVM, by default, will get 2.5G of heap during the installation. We don't advise changing heap sizes.

Understanding WebSphere ND

WebSphere ND is required for three components for HCL Connections 7: Deploy Manager and Node Agent on one side, and IBM HTTP Server on another.

Both Deploy Manager and Node Agent use 1G of heap by default. We do not recommend changing this.

IBM HTTP Server, on the other side, is not running as JVM.

Understanding Connections topologies

HCL Connections 7 installer lets you install HCL Connections 7 in three different topologies: small, medium and large.

Small means that all the apps will become the part of one single JVM cluster inside WebSphere with 2.5G of heap in total.

Medium will group all the apps in four different groups, which will reflect in four different JVMs with 2.5G of heap each.

Large will generate a cluster from each app, what would result in 21 JVMs with 2.5G of heap each.

Understanding HCL Connections 7 performance testing

All the numbers we are going to mention below are the result of running performance tests using HCL OneTest, the same tool that was previously used by IBM for giving sizing recommendations.

Tests were conducted for up to one hour of constant load with up to the number of parallel users using all main features of the application. As the underlying database, IBM DB2 v11.1 was used. All

tests were done on CentOS and RedHat Enterprise Linux v7 with IBM WebSphere Network Deployment 8.5.5.16 and 8.5.5.18.

All environments were tuned per the HCL Connections Tuning Guide.

HCL Connections 7 Sizing and Deployment Recommendations

Planning the Infrastructure

How to choose right instance or machine types?

HCL Connections should work without any issues on any type of infrastructure. It is designed to support most hybrid deployments. We performance tested scenarios that are going to be described below on big VMWare clusters, and the same stuff was tested on AWS and GCP public clouds. In all cases, depending on the number of users, you don't have to have the most expensive types of instances, but you shouldn't use those with a very small network bandwidth. The more you plan to grow regarding number of users, this will become more and more important.

The same goes with using physical machines. Ensure that each of them has enough network bandwidth for optimal operation of the cluster. The more users and data you are going to have can affect your performance. We suggest using at least 1G bonded network interfaces (if using physical machines) for a sake of high availability on a physical network level.

Planning local storage

So far, we haven’t seen a situation where we needed more then 150G of disk space per machine, no matter what the role and type of that machine was. Standard installation, in any case, shouldn't take more than 50G (including having installation archives unpacked to perform the installation) and proper processes should be in place to rotate the logs regularly and clean them up. In general, based on our observation during performance testing, on properly managed nodes, 100-150G of local disk storage should be enough for normal operation.

Planning shared storage

You will need NAS for shared storage, as one of the requirements for HCL Connections. The size of the NAS and available space depends on the number of users and the amount of data you want to put there.

For smaller installations, we would suggest at least 1T of disk space in RAID for a sake of data high availability.

For medium installations, we would suggest at least 10T of disk space in RAID.

For big installations, we would suggest at least 100T of available disk space in RAID, especially if Files application and file sharing are going to be used extensively.

Planning and sizing for load balancers

If you are using highly available environments, and if you are specifically going to have more than one IBM HTTP Server, you will need to have a load balancer in front of it.

If you are using physical deployments and you insist on physical load balancers, you can basically pick anything on the market which can handle the load that you are planning for. However, there is sometimes a cheaper option – small proxy instance with at least 4G of RAM and at least 2CPUs (one for system, one for network, since network is single threaded) and a good HA network interface and any software load balancer (HCL is supporting usage of both Nginx and Haproxy).

If you are on AWS or GCP (or any other public cloud for that sake) you can also go with whatever HTTPS load balancer is offered there, they should all be more then performant for what you need. But you can also spin up your own instance as just described above.

Planning and sizing ElasticSearch

HCL Connections 7 depends on ElasticSearch 7+ for QuickSearch and Metrics features. One way to get it up and running if you decide to set up the Component Pack (ElasticSearch 7 is a part of Component Pack in version 7).

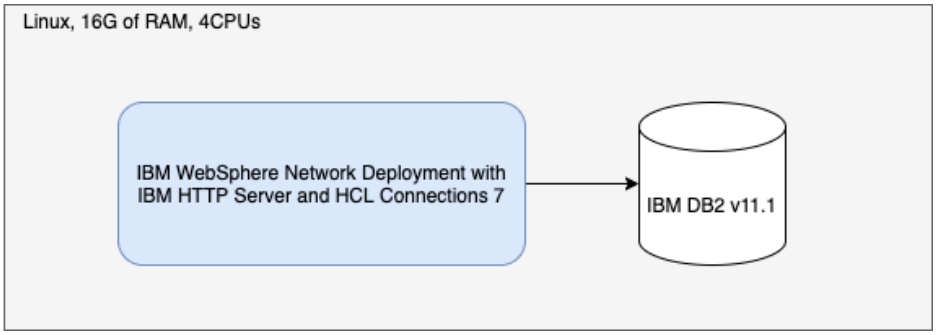
However, you can also decide to go and set up your own cluster – and that is fine. For that case, check out official ElasticSearch documentation to be able to plan it properly.

Small Internal Deployments

If you need a small deployment, either for demo/testing, or for a smaller number of users, you have multiple options. The number of users we are talking about bellow is a result of the tests we conducted. It should work for most of the situations, but of course it will depend also what you planning to do with your HCL Connections environment.

Connections-in-a-box deployment

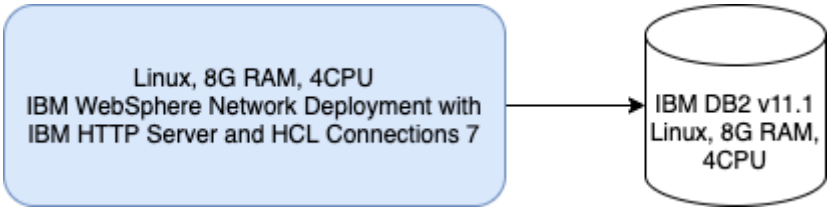
In case you want to try HCL Connections on one single machine, with a database collocated (in this case DB2 v11.1), you would need at least 16G of RAM and at least 4CPUs. This type of deployment would give DB2 enough memory for caching the data, and you will get more flexibility even going with a medium topology. While this machine size can handle its own collocated LDAP installation, we do not recommend using this type of topology for more than demo and development activities.



Connections with a small deployment topology

If you don't care about being high available, you can start with a Linux box with 8G of RAM and at least 4 CPUs. This would be enough to host IBM WebSphere ND and small deployment of HCL Connections while leaving enough space for the operating system. For running DB2 for example, you would need another machine with at least 8G of RAM and at least 4 CPUs.

This type of out of the box installation has proven to successfully serve up to 50 parallel users stressing the system, and we wouldn't advise this type of installation for more than 1000 users in total. However, we would not recommend using this type of deployment for any production environment. While this is more than enough for demos and testing environments, it could become quite hard to scale it once the number of users starts to grow as you would have just enough resources for running what you have.

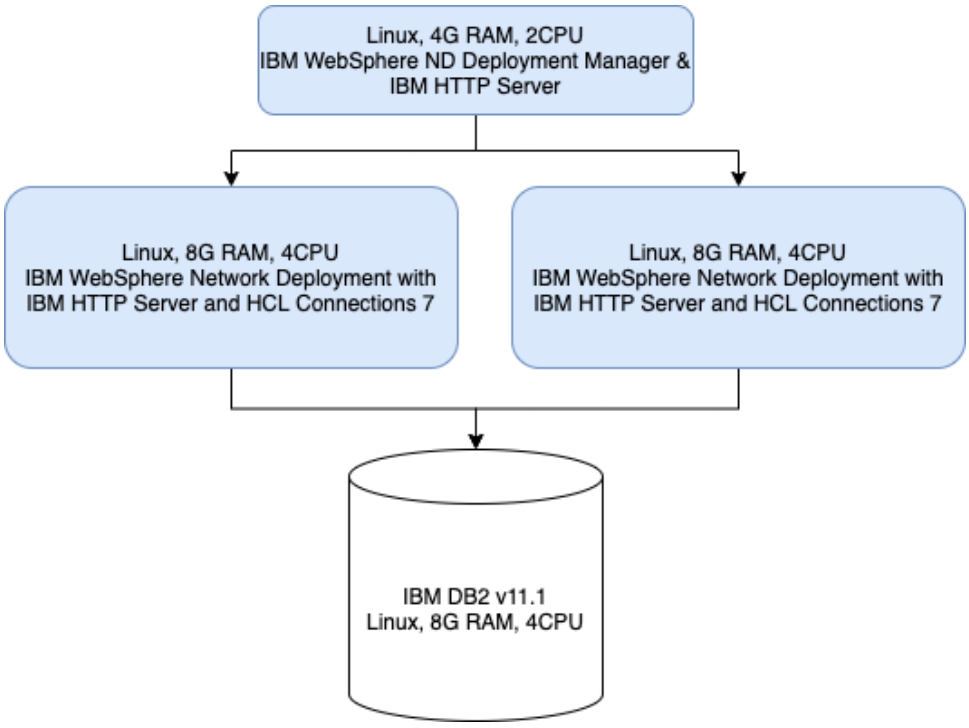


Small HA deployment

If you want your HCL Connections deployment to be highly available but to still stay at a bare minimum, we would recommend using four boxes for a deployment:

- One would have IBM WebSphere Deployment Manager collocated with IBM HTTP Server on a box with 4G of RAM and 2CPUs
- Two would host IBM WebSphere Node Agent with HCL Connections small deployment (single JVM) on a box with 8G of RAM and 4CPUs
- And one box would host IBM DB2 on a box with 8G of RAM and 4 CPUs

This type of deployment should easily handle, without big issues, up to even 100 parallel users, though we wouldn't recommend it for more than 600 users in total. However, with tuning it could be used to handle up to 1000 users in total.



Medium Production Grade Deployments

For production grade deployments you should start in general with medium type topologies and build (and tune) from there. We would not suggest non HA deployments here; however, we are aware that not everyone needs HA deployment.

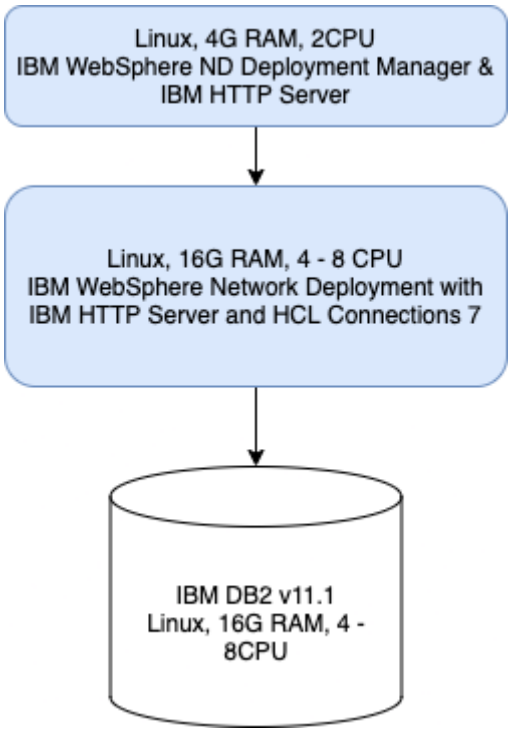
For medium deployment, we would in general recommend never to collocate your DB server with IBM WebSphere ND, and also to separate if possible IBM WebSphere ND Deployment Manager, IBM HTTP Server and IBM WebSphere ND Node Agent.

Non high available medium topology deployments

In this scenario, to handle up to 500 parallel users (or up to 20000 users in total) we would recommend going with:

- Single node with IBM WebSphere ND Deployment Manager and IBM HTTP Server with 4G of RAM and at least 2 CPUs. Please note that you need to tune your IBM HTTP Server which would, out of the box, be configured for up to 600 parallel connections.
- Single node IBM WebSphere ND Node Agent with HCL Connections 7. Here would be 5 JVMs in total – one for WebSphere Node Agent, and four for four different clusters. In total, Java would use 11G of RAM. However, our tests have shown that a box with 16G of RAM and 4 CPUs can handle 500 parallel users without latency going sky high.
- Single node IBM DB2 v11.1 with 16G of RAM and 4CPUs.

Note that in this case the question is also how many users, and how much data you plan to have. For DB server, it is always good to add more RAM and CPUs to be able to cache more in memory and not have any wait. However, with 32G of RAM and 8CPUs in we were able to run a couple of thousands of users in parallel without seeing any issue with server load.



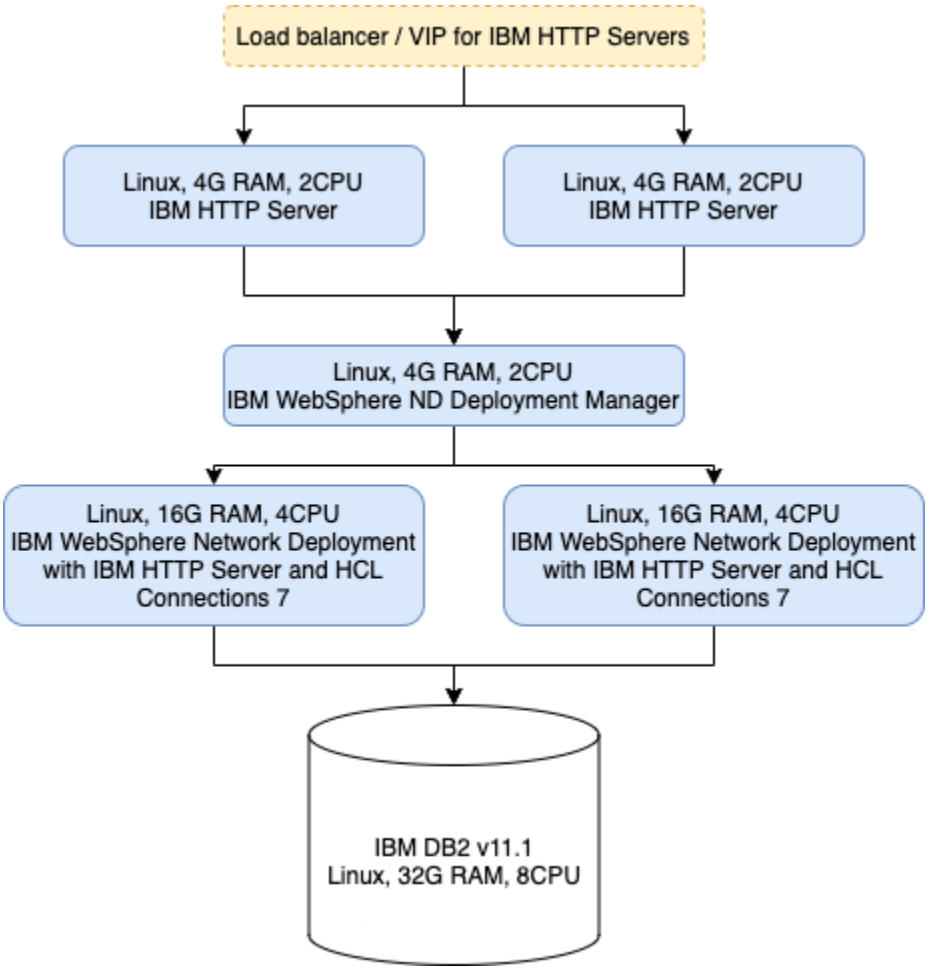
Highly available medium topology deployments

For this type of environment, we are looking to supporting 1000+ parallel users and 40.000+ users in total. For this type of load and data, we are going a step forward and recommending also making IHS servers high available and putting them behind some type of load balancer.

To support specifically scenario of 1000 parallel users hitting all the components, we would suggest:

- Two IBM HTTP Servers behind the load balancer, each having 4G of RAM and 2CPUs.
- A single server with IBM WebSphere ND Deployment Manager, 4G of RAM and 2CPUs
- Two IBM WebSphere ND Node Agent servers with HCL Connections 7, each having 16G of RAM and 4 CPUs
- At least single node IBM DB2 v11.1 with 32G of RAM and 8 CPUs.

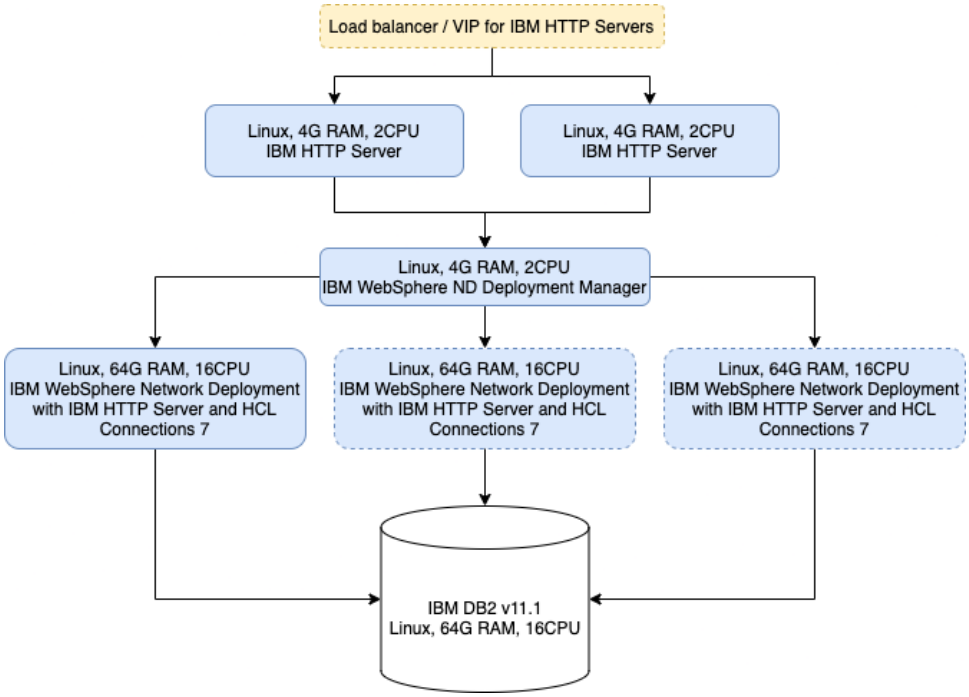
This type of deployment can support ***at least*** 1000 parallel users. It is possible to push even more, but the real question here is what are you doing, and what do you want to do. Also, without changing anything sizing related, this type of environment gives you a flexibility to eventually reconfigure your HCL Connections layout and, if you have an app which is utilized specifically hard, to move it to its own cluster. However, we would advise to stay with four clusters, and eventually play with what goes into which of four JVMs.



Large topology production deployments

Large topology means that each app will become its own JVM. Each JVM will, by default, get 2.5G of heap, which in the end gets us to machine with at least 64G of RAM and 16CPUs for IBM WebSphere ND and HCL Connections 7. We would not advise going with a larger machine then this.

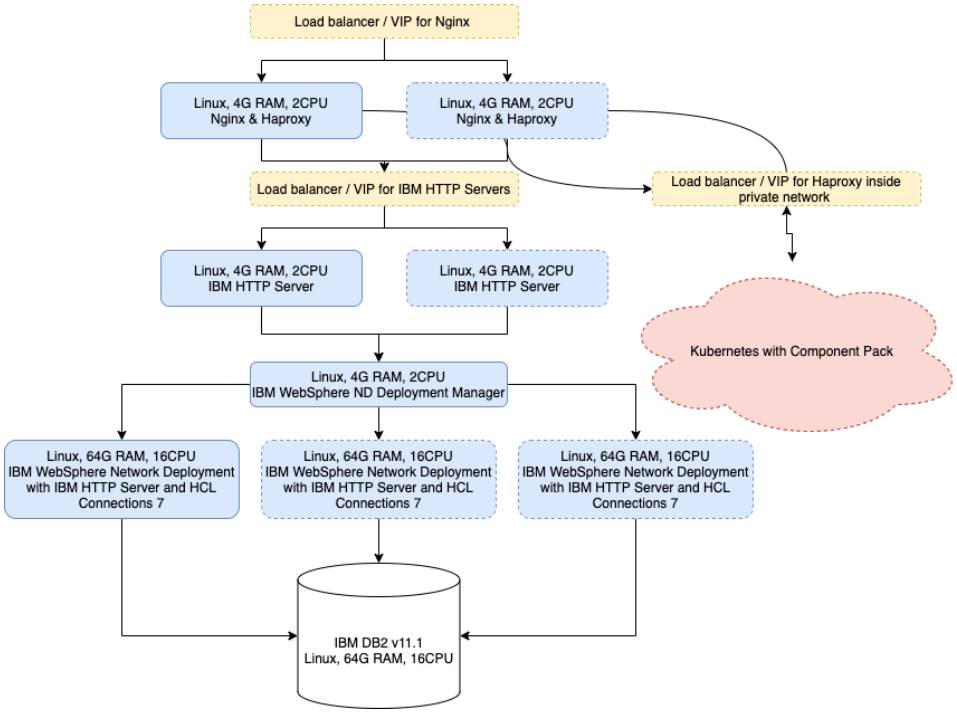
This type of deployment is definitely not suitable for a single node scenario. Real life scenarios show that customers who don't need HA use a single such machine to handle 250.000+ users. Of course, this again depends on what exactly are your trying to do, but in general, to go really large, we suggest scaling horizontally and properly tunning your environments.



Understanding HCL Connections 7 with Component Pack

Component Pack for HCL Connections is a set of extensions for HCL Connections which is running exclusively on Kubernetes. If you are going to use HCL Connections with Component Pack installed, it is possible to go different ways and use different topologies to set it up. However, two extras that you get on any of the environments that was described above, are:

- One extra proxy server which would be also the entry point for your whole HCL Connections environment. We suggest here a small Linux machine (4G of RAM and 2CPU) with Nginx & Haproxy. Of course, if you are going all the way HA, that machine as well needs to be duplicated, and some load balancer needs to be put in front of it. To save on resources, we are collocating here Nginx as (reverse) proxy and Haproxy which is going to act as a control plane for your Kubernetes cluster. All the communication between Connections and Component Pack is going to go through Haproxy or the load balancer in front of it in case you are going HA here is well. On another side, all the applications from inside the Kubernetes cluster will talk directly you your IBM HTTP Server or to the load balancer in case you have more than one IBM HTTP Server.
- Kubernetes cluster. It can be one single all-in-one node, or full blown, fully distributed Kubernetes cluster (or AWS EKS, RedHat OpenShift running on prem or in public cloud, or... basically any type of Kubernetes). At a bare minimum, in case you want to go with a single node all in on Kubernetes environment, you would need a machine with at least 8 CPUs and 32G of RAM.



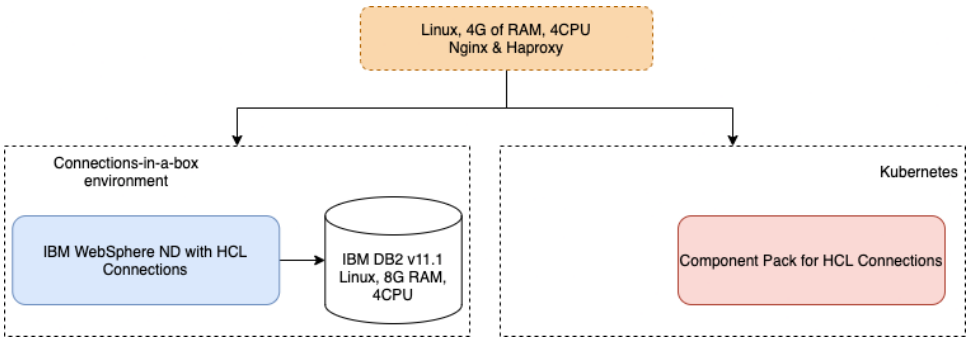
Sizing the network for HCL Connections with Component Pack

Kubernetes and the rest of HCL Connections need to be in the same local network. This means that you need to ensure enough bandwidth for the interservice communication. Otherwise, you are going to have serious performance issues.

HCL Connections and Component Pack basic deployments

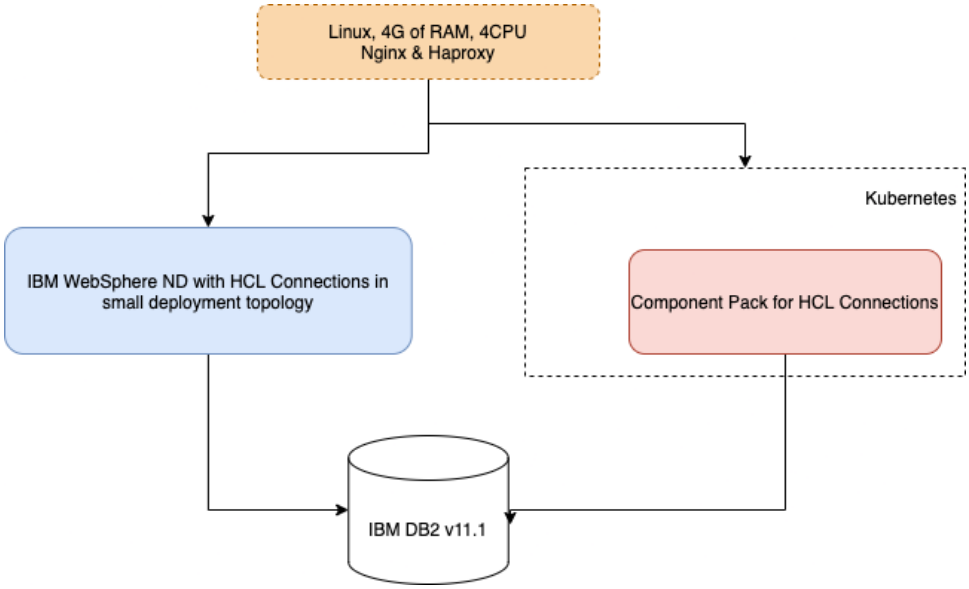
The most basic deployment of HCL Connections and Component Pack would be three machines:

- One with 4G of RAM and 4CPU for proxy (Nginx and Haproxy)
- One all in one HCL Connections environment as described in the section "Connections-in-a-box deployment"
- Single Linux box with 32G of RAM and 8 CPUs at least for Kubernetes and Component Pack.

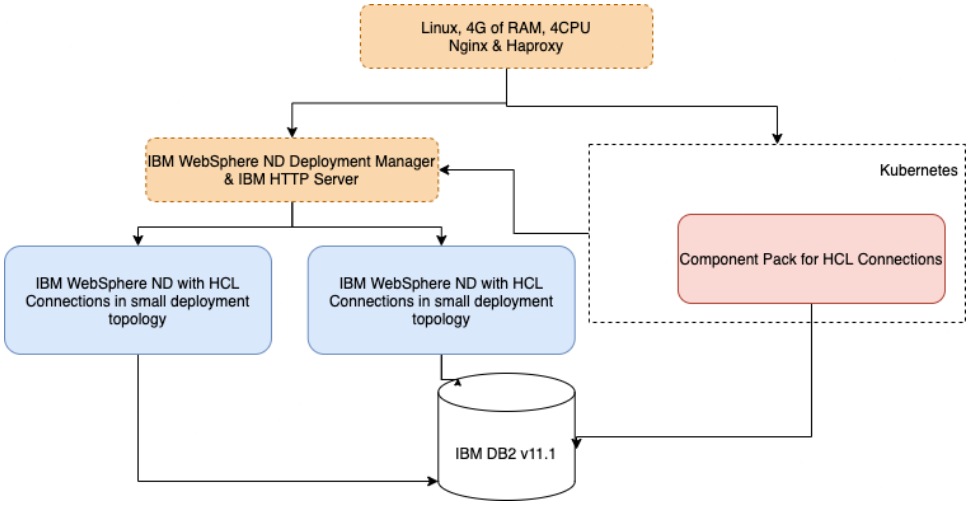


The same sizing limits mentioned in a section "Connections-in-a-box deployment" apply also to this type of environment.

Another option would be exactly the same as the previous one, where instead of using "Connections-in-a-box deployment" approach we would use "Connections with a small deployment topology" approach. It would look like this:



And last but not least, you could also go with a small HA deployment here. Everything stays the same as described in a "Small HA deployment" section, but there is that extra proxy and Kubernetes with CP:



In all those cases, this single node Kubernetes cluster with the Component Pack on it is limited by what the rest of HCL Connections can serve, which means that all the sizing limits that apply on small internal deployments without Component Pack also apply when Component Pack is there.

Production deployments of HCL Connections with Component Pack

All the data we have for the HCL Connections so far applies here as well. However, this is the point where Kubernetes cluster (and Component Pack inside of it) needs also to be able to follow what HCL Connections side can do. Before you proceed reading further, check out what you need to think about when deciding on how to size Kubernetes cluster for Component Pack.

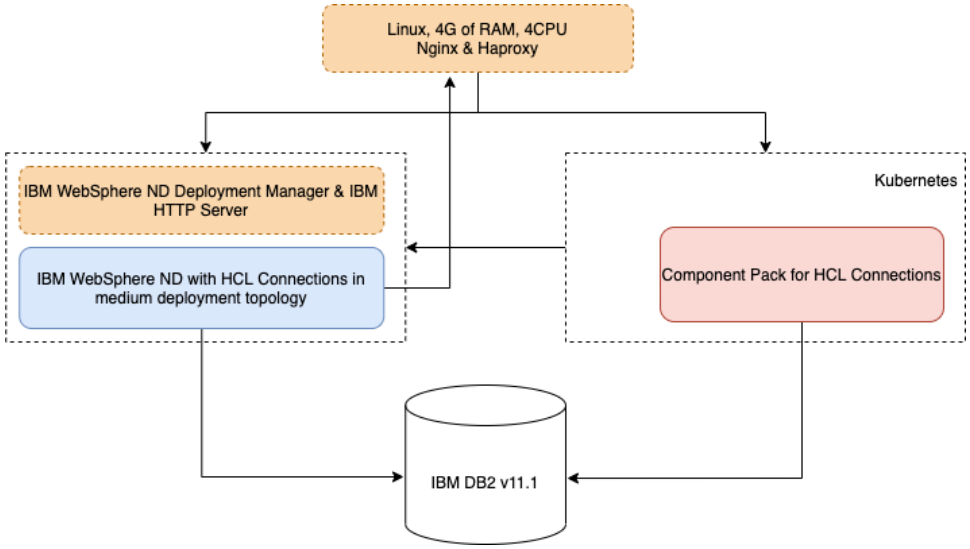
Small production grade clusters with Component Pack

The most basic example of a small HCL Connections installation with Component Pack included would be:

- Single node proxy server with Nginx and Haproxy (Linux box, 4CPUs and 4G of RAM)

- Single node all in one IBM WebSphere and HCL Connections installation (Linux box with 32G of RAM and at least 8 CPUs). Here we would go with a medium topology during the installation.
- Single node all in on Kubernetes with Component Pack (Linux box with at least 32G of RAM and at least 8 CPUs)
- Single node with DB server, IBM TDI and eventually LDAP collocated (16G of RAM at least with 4-8 cores, depending on the type of load you are planning to have; the rule of thumb here is that more you plan to have to do with Files app, more cores you want).

This type of environment is benchmarked for up to 200 parallel users and can be tuned for more. However, scaling this properly would be complicated, so we don't advise using this for more than 5000 users in total or if you plan to have load of more than 200 concurrent users.



This is ideal, and widely used also by HCL, for a preview type environment. Such environment powers preview.hclconnections.com and environments given to the customers to test HCL Connections themselves.

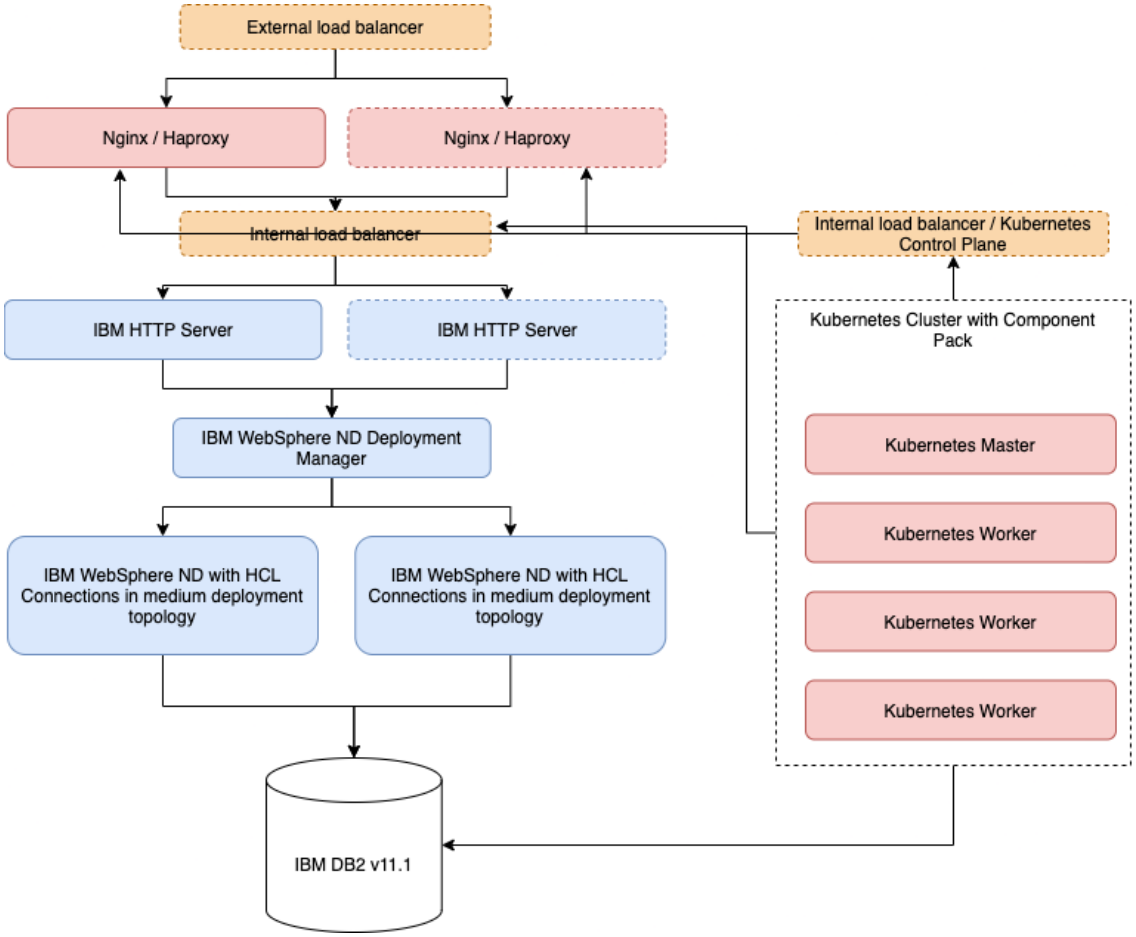
Medium production grade clusters with Component Pack for up to 1000 concurrent users

Out of the box, considering that you made the decisions on how to go with your Kubernetes cluster and decided to save on the number of Kubernetes nodes, on top of how would "Highly available medium topology" layout look like, you need also:

- At least one proxy server with Nginx and Haproxy (at least 4G of RAM with at least 2 CPUs). If you already decided to go HA for IBM HTTP Servers, then you need at least one more server like this and the load balancer in front of those two.
- One Kubernetes master (at least 8G of RAM and at least 2 CPUs)
- Three Kubernetes workers (each with 32G of RAM and 8 CPUs)

Out of the box installation of Component Pack will spin up three replicas for each pod you have. This is already the point where, if you need to support more concurrent users, you don't have to add more resources in your infrastructure, but need to understand what is happening and how to scale specific pods depending on your load.

We also don't recommend going with more beefy worker nodes, but in case you really need more resources here, it would make sense to add more workers and scale Kubernetes horizontally. However, Kubernetes scaling strategies are out of the scope of this document.

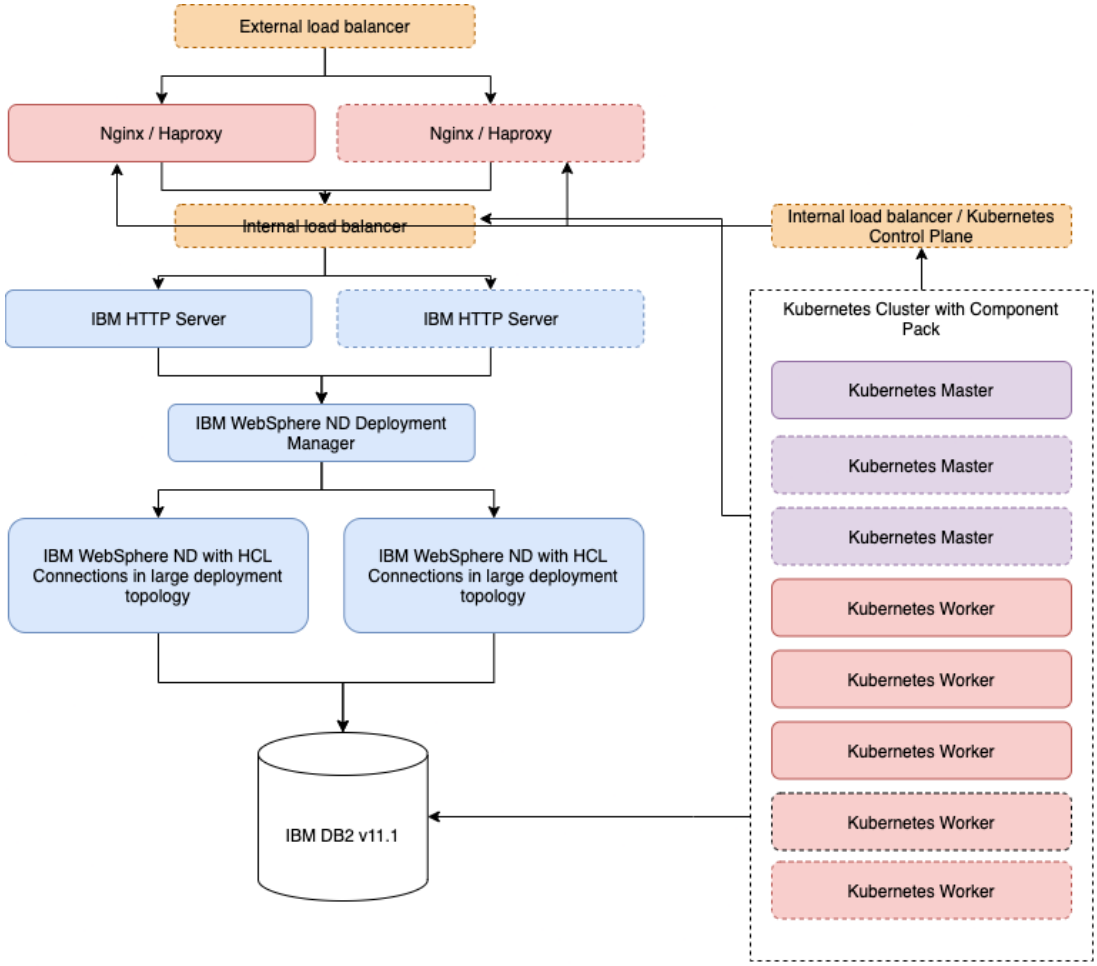


Large production grade clusters with Component Pack

For the large production grade clusters, we would suggest nothing less than what is already described in "Large topology production deployments" section. However, on top of that, you would need to properly size and tune your Kubernetes cluster. The delta here, when it comes to Kubernetes, would be that you would need to add at least three instead of one single Kubernetes master node, and start scaling your workers. However, scaling workers doesn't have to happen based on any multiplication factor. It really just depends on what is the shape of your traffic and what exactly needs to be scaled. Properly monitoring Kubernetes can give you exactly those answers. However, scaling and monitoring of Kubernetes clusters is not in the scope of this document.

Official recommendation for large deployments (we consider large everything from 200.000+ users in total) is:

- Large topology production deployment as described previously.
- Two frontend proxies for Nginx and Haproxy collocated, at least 4G of RAM and 4 CPUs each (remember that network is CPU bound; if you are going to increase network traffic based on the load coming from a big number of users, you want to be sure that you have enough CPU cores).
- Front load balancer in front of those proxy servers.
- At least three Kubernetes masters for the sake of HA and better load distribution on Kubernetes itself.
- At least three Kubernetes workers. However, start scaling here with adding more workers depending on the type of traffic you are going to have.



Overview of Sizing Recommendations

We are going to wrap up the process of deciding what is best for you either, if you are deciding based on concurrency in peak periods of time, or if you are deciding based on projected total number of users.

To save some money and time, there are some general rules of thumb here:

- You don't need HA if you are not going to production.
- If you don't need HA in production that's fine. We have customers handling 250.000 users in total on a single node with large topology deployment tuned for their cases.

Deciding based on total number of projected users without Component Pack

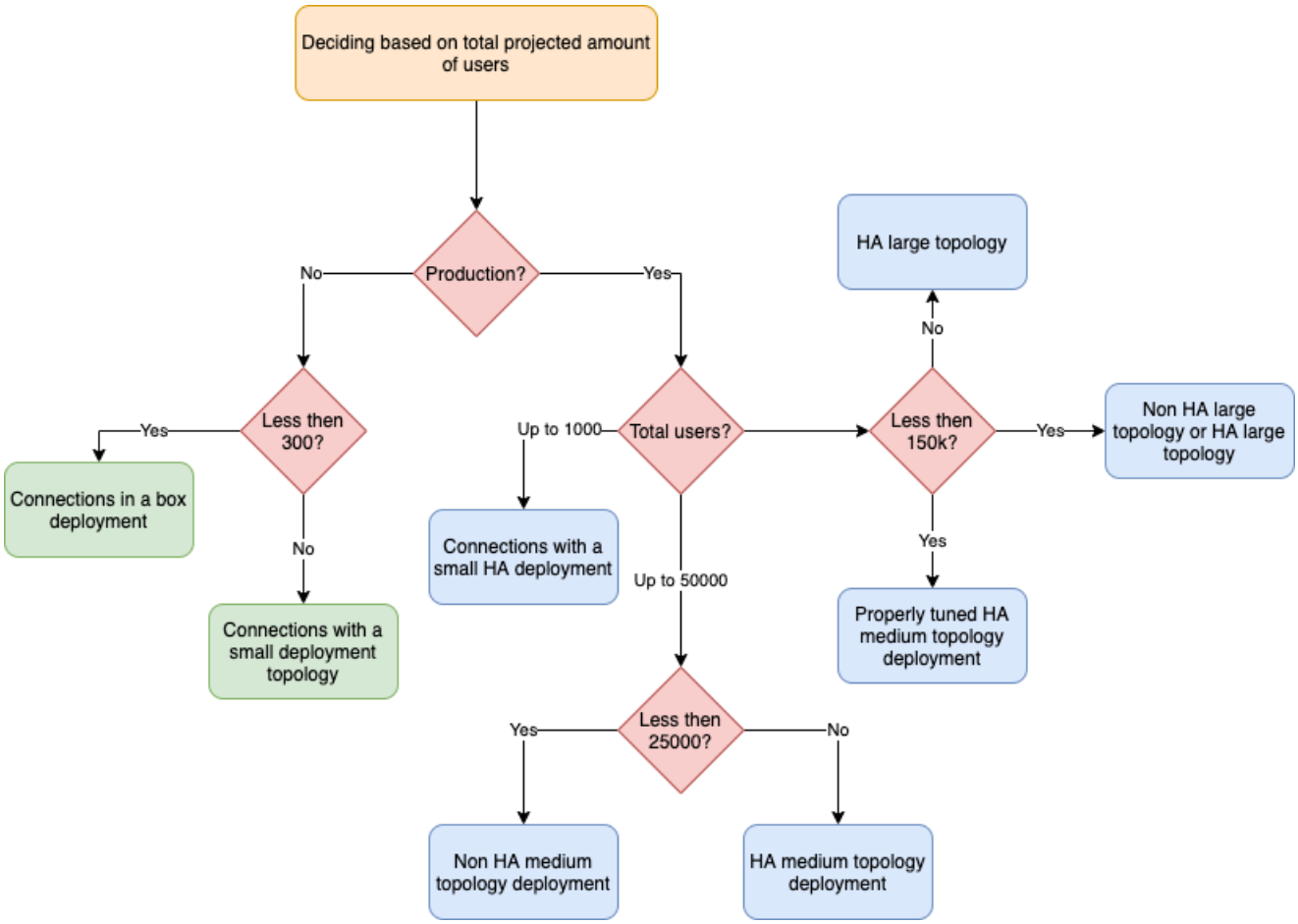
The important factor here is that, if you decided to go without Component Pack, you are saving on proxy servers, additional load balancers and the whole Kubernetes cluster.

Each of deployments mentioned bellow are already covered previously in the text. However, be aware that it is impossible to calculate exactly for each type of use case that customer can have.

What is important to understand is:

- For non-production scenarios, small topologies are fine in general, and you don't require HA.
- For production scenarios, you are deciding between:
    - HA or no HA

- Medium vs large topology deployments depending on the number of users and expected load.



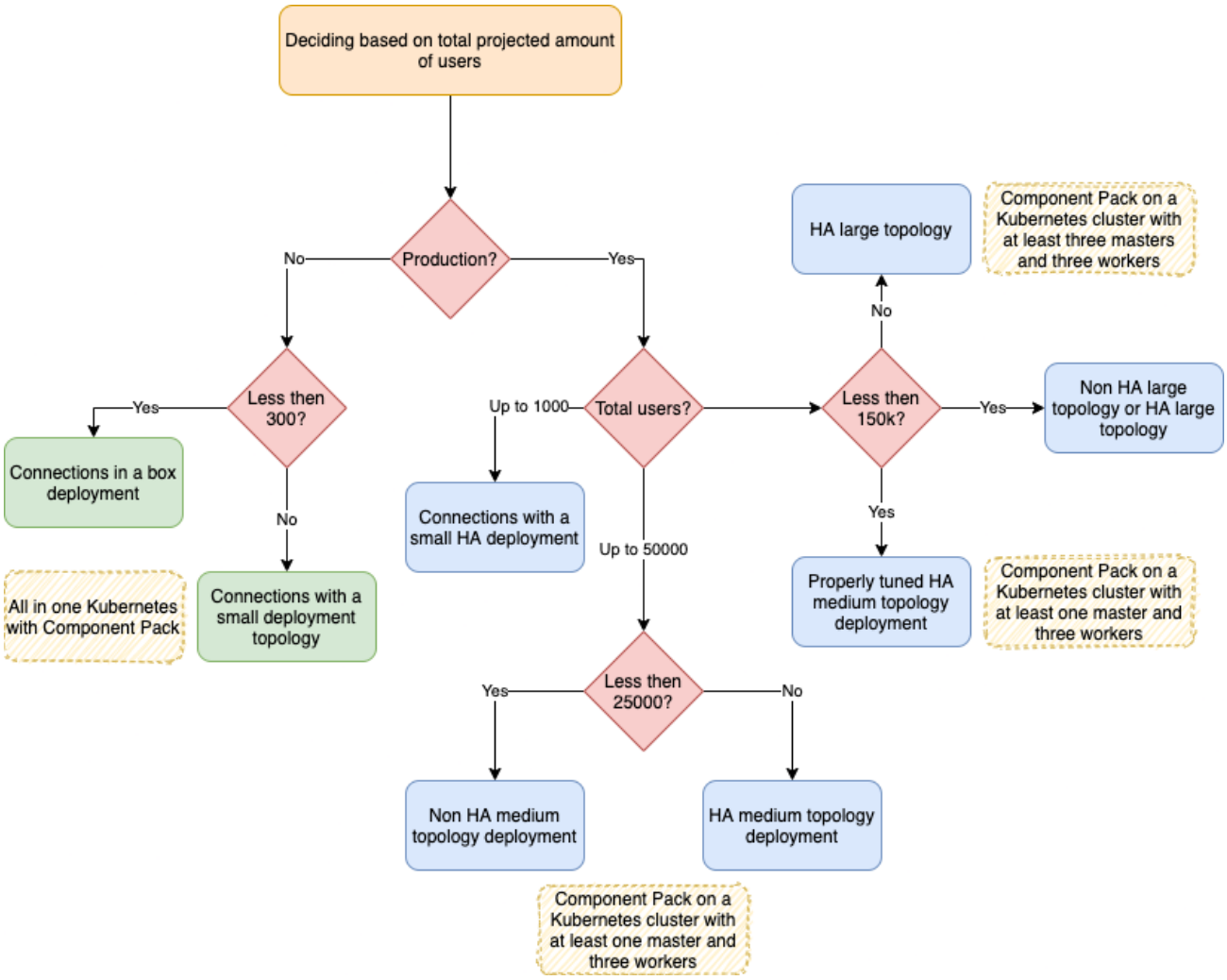
Deciding based on a total number of projected users with Component Pack

Note that here, for a sake of easier visualization, we added the recommendations for the minimum Kubernetes setup needed for specific types of deployments.

What is really important to understand here is that Kubernetes has its own scaling and sizing rules, but in general:

- For development/PoC/non-HA scenarios single node / all in one Kubernetes with Component Pack is just enough. However, note that setting up Kubernetes cluster is different for setting up single node and non-single node (with at least one master and one worker). This is important to understand as moving from single to multiple nodes means also recreating your Kubernetes environment and reinstalling Component Pack.
- For production scenarios you should never need more than three Kubernetes masters sized as described above. Per official Kubernetes documentation this should be enough for up to 100 workers, which I am sure you will never get to with Component Pack alone.
- Once you get to the large topology, you should, in worst cases (worst – having more than 250.000 users in total) consider scaling horizontally your WAS nodes (adding one or two more, depending what is your load and number of users). However, before adding any more infrastructure, it is important to tune everything that can be tuned.
- Kubernetes workers should work the way they are described already. There is no gain on adding more CPU/RAM in them, as the real gain would be reached by scaling them

horizontally (adding more workers). However, we do not predict that you will ever need more than six workers in total, even in the biggest possible environments.



Deciding based on the needed concurrency with Component Pack

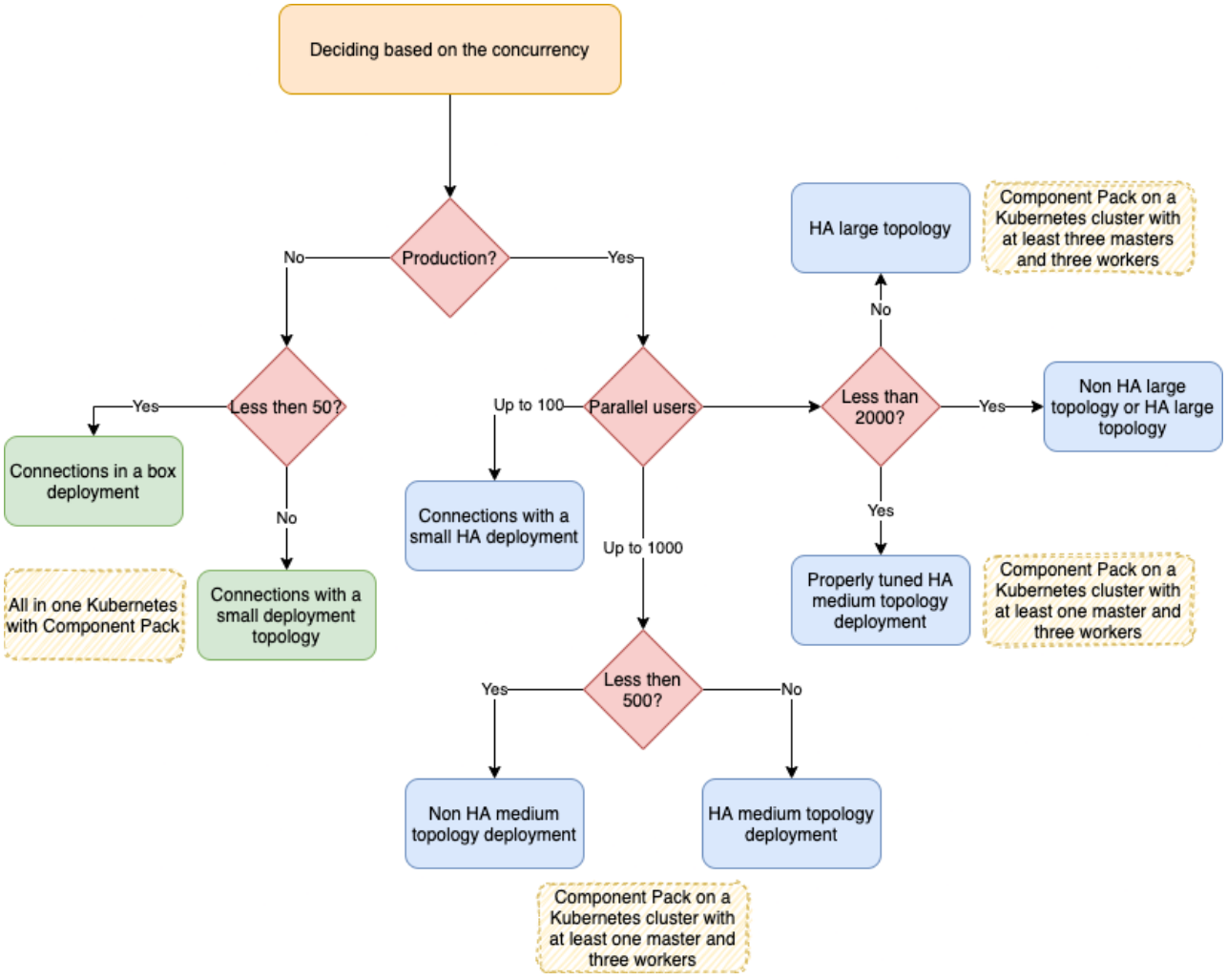
The same rules as above apply here as well.

From our performance testing, it is important to understand relations as well:

- Environments that we tested with ~50 users stressing it in parallel were handling no more than 1000 users in total. The most important difference in performance here was if it was all in a box or app and DB separated (as expected). We also have a production customer using this "Connections with a small deployment topology" in production considering that that customer doesn't have more than a couple of hundreds of employees in total.
- Environments we tested with up to 100 parallel users that were really stressed were small HA deployments. This translates to a customer having up to ~5000-10000 users. However, we are trying to be quite conservative here.
- When talking about up to 1000 parallel users this translates to our customers which have around 50.000 users in total that heavily use their Connections environments. It is important to distinguish here at around this number it is not anymore, as stated before, about adding more resources, but about starting to tune.
- In the area between 1000 and 2000 parallel users (which translates to 50k – 100k and more users in total) it is all about tuning and making right decisions. Tuning here is also deciding to

go with a large topology instead of medium, which means bigger machine(s) to satisfy the bigger amount of JVMs.

- From 2000+ parallel users and up it is all about making decisions if to go with HA or not and tune tune tune. There are no magic numbers that can exactly, for each and every customer, give proper sizings. For example, when Customizer (just one of the Component Pack components) is used or not. It depends a lot on what is going to be done.



License

Disclaimer