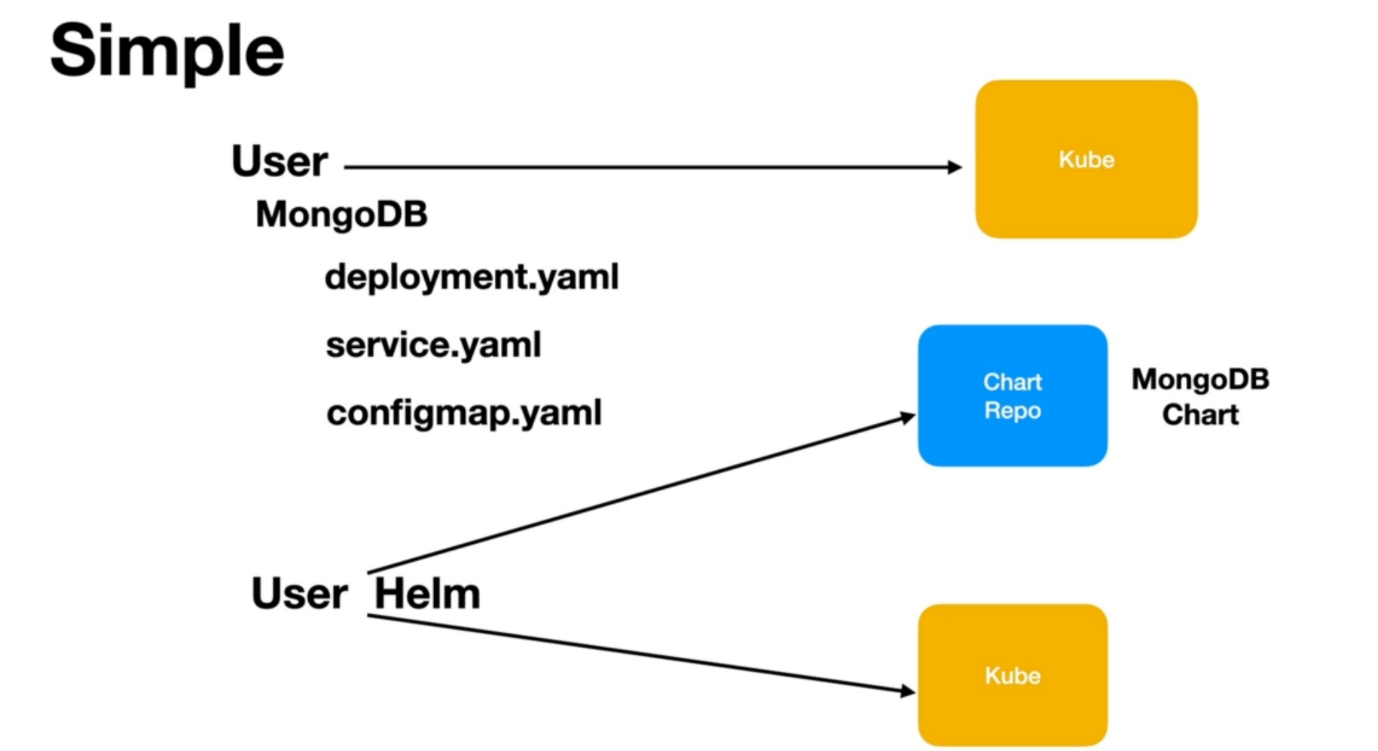
**7. After Helm**

* in this, we will learn why should we use helm and what is the advantages of helm.
* Helm simplifies the Kubernetes deployment process by abstracting out all the complexity.

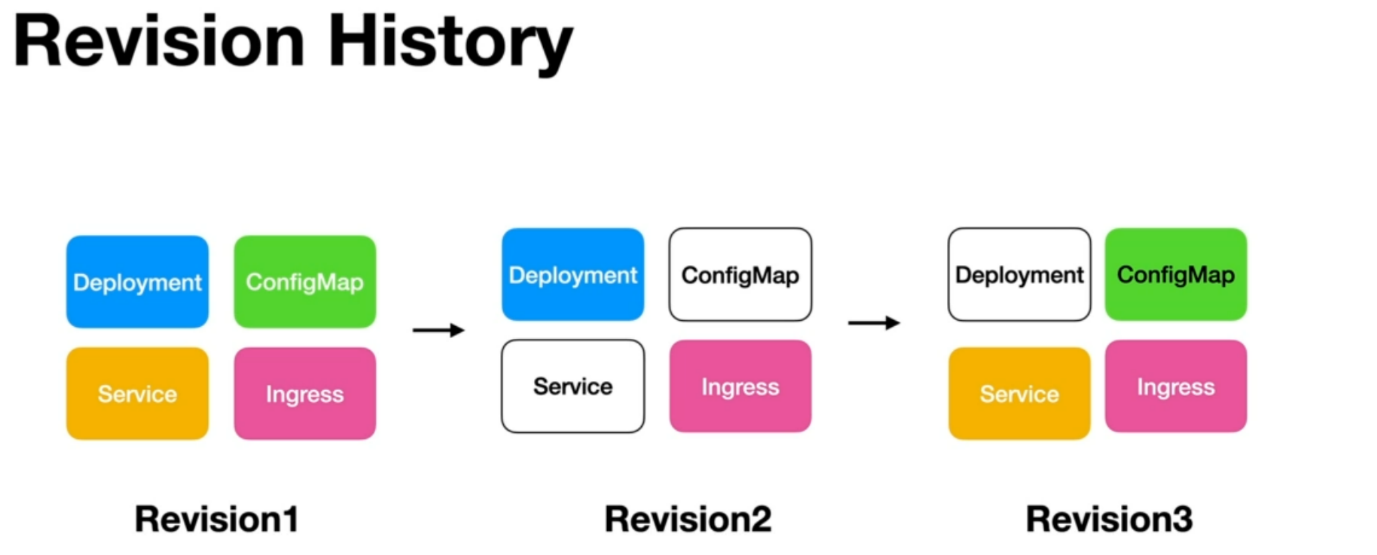
**Mangodb**



* we as a DevOps engineer, we are asked to install MongoDB application to the Kubernetes cluster, will have to understand and create all the resources that are required to configure containers, networking, volumes, etc., which can be overwhelming. We then hand over these files to the kubernetes API server, which will create the resources for us.
* Using helm. All this can be accomplished with a single command, helm will pull a MongoDB chart from a centralized chart repository and then it will do the installation for us with a single command.
* This MongoDB chart will have all these template files already in it, and it comes with default set of values which we can override if required.
* The MongoDB experts or the MongoDB chart Experts create these charts for us.

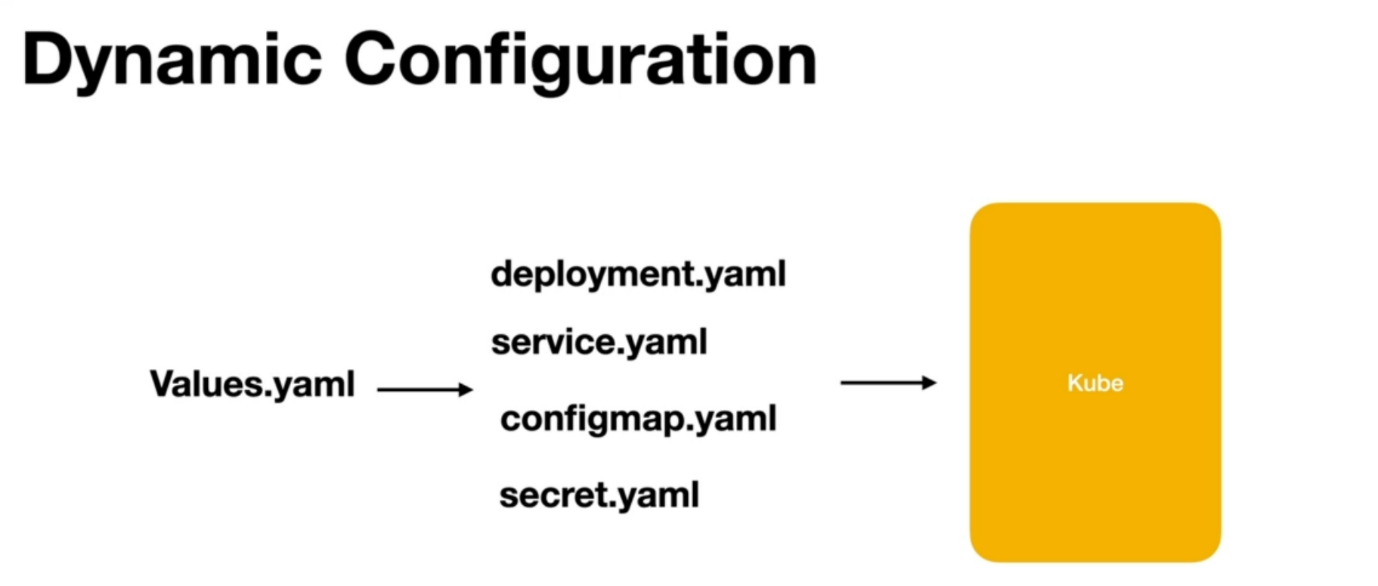
--- **note** - If we are working with MySQL Database, the MySQL experts will create these charts for us. We do not need in-depth knowledge or understanding of how MongoDB works or how MySQL works to install that application to the kubernetes cluster.

**Helm revision history**



* Helm maintains a revision history as we do installations and upgrades.
* Helm will create a revision for us and it will store all the templates and configuration information used for that installation or upgrade to a particular location.
* as you do upgrades, it will create another revision for it. For example, we have changed the service and config map. Now we want to update. We want to upgrade whatever is there on the Kubernetes cluster. You simply do an upgrade.
* You do not have to do the complete installation again as you do that upgrade and will create revision2 it will store all the new templates or the changed templates to the location as a revision2 and then you change the deployment another revision will be created.
* If you want to revert to a particular revision, we can easily do that because Helm maintains the revision history and all the templates and configuration used for a particular release, installation, or upgradation.

**Dynamic configuration**

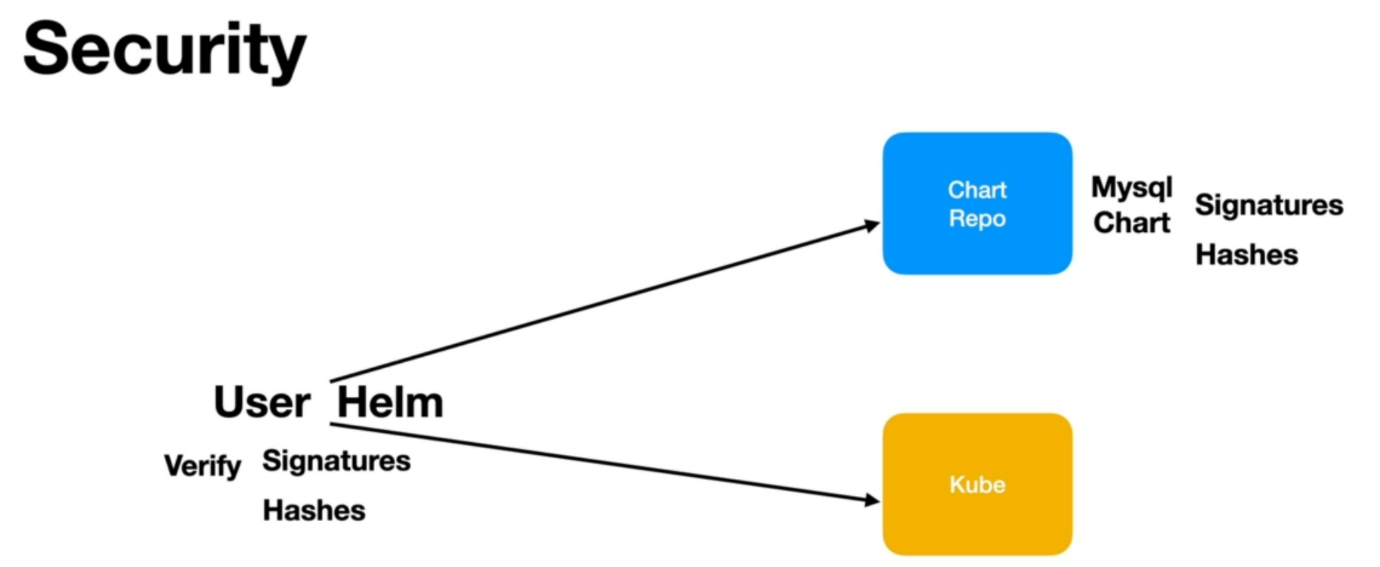


* when we work with Kubernetes directly to do our deployments will have to create all the yaml files, which are static. They do not accept any values. we cannot pass any parameters dynamically to these static yam files.
* we repeat this across upgrades and, we repeat these files across projects. Thanks to helm. Helm gives us all these files as templates when we create a chart. All these files are generated and they will have placeholders for parameters.
* Helm uses another file called Values.yaml, using which we can pass parameters to this template files.
* The combination of these templates, values.yml will be used by Helm to generate the final yaml that will be sent to Kubernetes.

**Intelligent deployment**

* Helm is very intelligent when it comes to deployments, when we directly work with Kubernetes, we will have to mention the order in which the resources should be created. Example config map secrets should be usually created before deployment and services and we need to provide this information to Kubernetes but Helm is intelligent.
* It knows the order in which Kubernetes resources should be created and it will automatically do it

**Security**



* Helm has inbuilt support to ensure that the charts that are downloaded from the central repositories are secured.
* the charts can be signed using cryptography and hashes can be generated and when we install these charts, we are pulling them from the central repos.
* Helm will verify that these charts are really from the source we are expecting and they are not tweaked by any hacker.