It is recommended to take a few steps to convert your monolithic app to microservices using Kubernetes. This is a high level overview of the entire process.

1. **Decompose Monolith**: Analyze and identify the different functional components of your monolithic application. Determine how to split these components up into individual microservices, based on the responsibilities of each component.
2. **Define service boundaries:** Establish boundaries between microservices in order to ensure loose coupling, independent scaling and independence. Determine the APIs between services and the data contracts.
3. **Containerize microservices**: Containerize every microservice using Docker. Create Docker images that contain the application code, dependencies and runtime environment for each microservice.
4. **Orchestrate Kubernetes:** Set up a Kubernetes Cluster or use a managed Kubernetes Service like Amazon EKS or Google Kubernetes Engine or Azure Kubernetes Service. Deploy containerized microservices into the cluster.
5. **Load Balancing and Service Discovery:** Use Kubernetes features such as Services and Ingress to provide load balancing and service discovery for your microservices. Ingress allows external access to each microservice, while Services exposes them internally within the cluster.
6. **Configure & Manage Networking:** Define networking policy to control communication between services and ensure proper isolation and security. Use Kubernetes Network Policy to restrict traffic between services according to specific rules.
7. **Manage Microservices Lifecycle:** Use Kubernetes deployment features such as Deployments or StatefulSets to manage your microservices lifecycle. This includes rolling updates and automatic recovery.
8. **Implement Service Monitor:** Setup monitoring and logging of your microservices. Use Kubernetes native monitoring solutions such as Prometheus or Grafana, or integrate third-party monitoring tools.
9. **Implement Service Resistance:** Implement fault-tolerance mechanisms such as circuit breakers and retries within your microservices. Use Kubernetes features such as health checks and readiness probing to ensure service availability.
10. **Continuously deploy and scale:** Setup a CI/CD pipe to automate deployment and scaling your microservices using Kubernetes. Implement techniques like Blue/Green or Canary deployments to control rollouts.

Remember that the transition from a monolithic to a microservices architecture requires not only technical but also cultural and organization shifts. It's important to involve both your development and operations team throughout the process, and adapt your development practice to align with the microservices and Kubernetes principles.

**Tools used for provisioning and configuring the databases:**

There are a number of options to configure databases using Kubernetes. Some popular choices include:

1. **Helm:** The Helm package manager allows users to manage and install applications. Charts: These are packages which include the necessary resources for configuring and deploying a database.
2. **Kustomize:** Kustomize allows you to customise database deployments including environment variables and secret variables.
3. **Kubernetes Operator:** Kubernetes Controllers are custom controllers which extend Kubernetes functionality to manage complex apps. Each database vendor has their own Kubernetes operators that simplify the deployment and management of their databases. These Operators offer a high level of abstraction allowing the user to define custom resource and manage their databases with familiar Kubernetes GUIs.
4. **KubeDB:** KubeDB allows you to run production-ready databases with Kubernetes. KubeDB can support different databases such as MySQL PostgreSQL, Elasticsearch, Redis and others. KubeDB also supports the use of declarative YAML setting. KubeDB makes it easy for users to deploy, configure and manage databases.
5. **Tools and extensions for specific databases:** Many databases offer tools or extensions that are tailored specifically to Kubernetes. You can, for example, use pgBackRest to backup PostgreSQL and Percona XtraDB Cluster cluster MySQL. Kubernetes' StatefulSets are used to manage applications that require state, such as MongoDB.

Consider all the options you have based on your needs and wants.