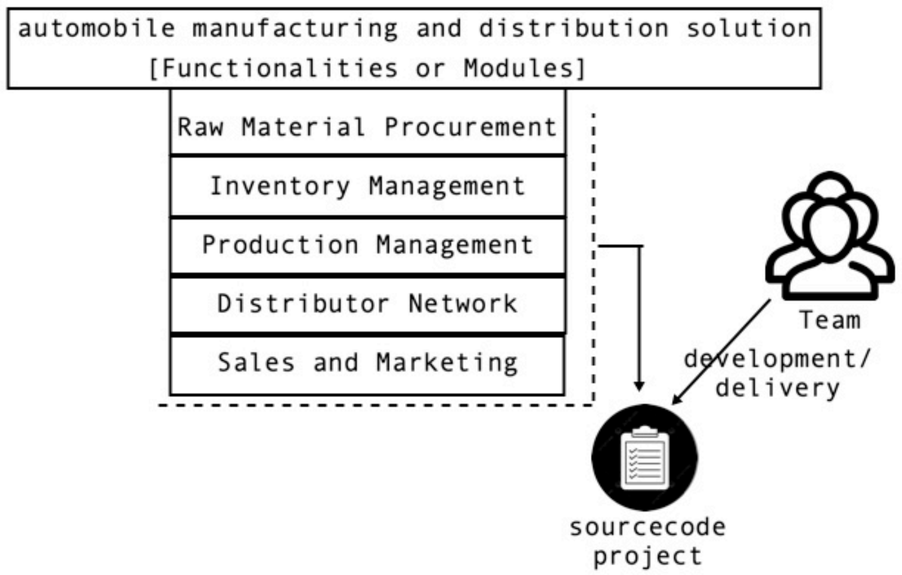
Monolithic Application Architecture:



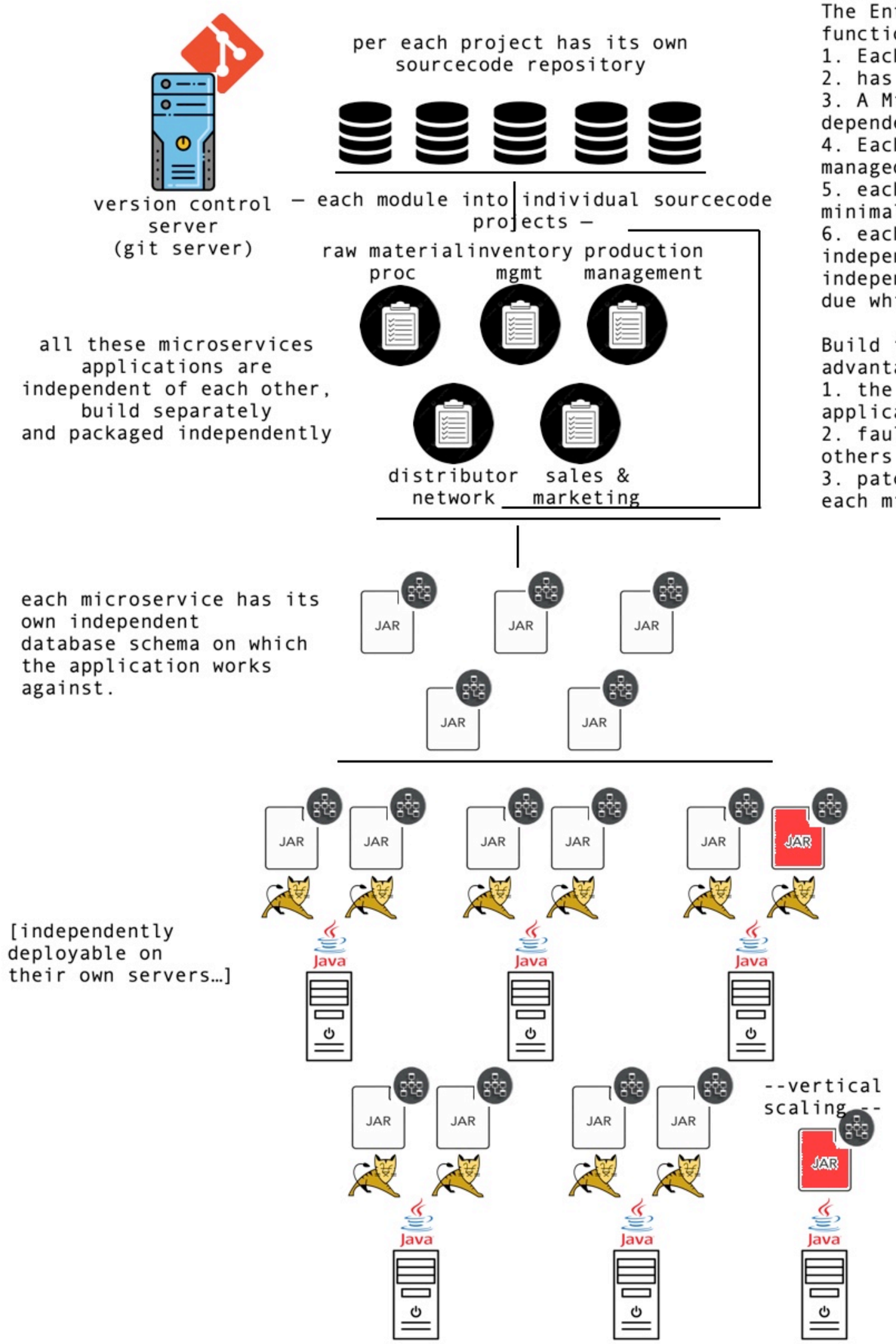
When we are building the project based on monolithic application architecture, the whole system is built out of one single sourcecode project and it is being developed by only one team.
Since the project is very huge in size, we need a bigger team for development/delivery of the application. There are lot of challenges in managing the bigger team like tracking, monitoring etc

Instead we can break the members of the team into smaller groups as sub-teams each working on one functional area or module, so that man-power management becomes easy. But there are lot of challenges in managing several sub-teams
1. when a sub-team has made any changes in one of the module of the project, it would effect all the modules of the system and requires coordination and planning to ensure other module teams are aware of these changes and should be prepared to make necessary changes from their end to make their modules compatible

2. even a sub-team has finished their work, they cannot release their code independently as the whole system is built into one single deployable artifact. The team has to wait for rest of the modules/sub-teams to finish their work which means the entire team sit Idle

By the above we can understand the monolithic application architecture based applications doesnt support parallel application development and takes lot of time and cost in development/delivery.

Microservices Application Architecture



Microservices Architecture
The Enterprise large-scale complex business application that comprises of several functionalities are broken down into smaller microservices modules/applications.
1. Each Microservice project is built into its own sourceocode project
2. has its own sourcecode repository
3. A Microservice module/project will not have other microservice modules/project as dependency between them interms of ClassPath References
4. Each Microservice has their own database schema against which the data is stored/ managed.
5. each microservice project can be developed by its own independent team and requires minimal coordination interms of planning the features across
6. each team can work on their own Sprints, planning and release management activities independent of other teams which mean self / autonomous teams that works in parallel and independent of others
due which we can achieve agility in development and delivery of the application

Build independently and deployed isolated from other microservices:
advantages:
1. the system resources allocated to a microservice is dedicated to that service/ application only so that the load on one microservice will not affect the other
2. fault isolation, if there is crash in one microservice, there would not be any impact on others
3. patching, upgrading and release management activities can be carried independently for each microservice without affecting the down-time or availability of the others