Compute decision guide

Not sure what model of compute best supports the app or service you are building? Here are some helpful guidelines to point you in the right direction.

IBM **Bluemix**

	OpenWhisk	CF App Runtimes	Containers	Virtual Servers	On-prem Datacenter
Workload characteristics (sweet spot)	Stateless/shortliving Written in a well-defined set of languages	Stateless Http(s)/websockets	Longer-living Any protocol Custom OS binaries required	OS customizations Full OS control Stronger isolation requirements	Special HW required Compliance regulated
Workload examples (sweet spot)	API/microservice/web app implementations Mobile backends Reaction to streaming / data IoT, Cognitive, etc. events	High-volume web apps/APIs	Continously runnning processes (e.g. game engines) Distributed technologies (e.g. mongodb, zookeeper)	Apps having special OS requirements Apps packaged into existing VM images Live video streams (resource heavy)	Data which must be in on-prem data center Mainframe apps
Time to provision	Milliseconds	Seconds/Minutes	Seconds/Minutes	Minutes	Weeks/months
Utilization	Highest	Higher	Higher	High	Low
Ability to reuse existing apps	Low	Lower	Medium	High	Highest
Charging granularity	Blocks of ms execution time	Hours	Hours	Hours	CapEx
Developer view	Just the app code	Just the app code	Container	Virtual server	None
Autoscaling	Inherent, no delay	Mgmt function	Mgmt function	Mgmt function	None
Artifact	Action code, trigger, rule	App code	Container	Virtual server	Physical machine
Developer usage	Uploads only artifacts No explicit management of computing resources required. No starting and stopping of application required.	Uploads complete application using a CF supported runtime. Explicitly binds services to application Explicitly starts/stops the cloud application. Entire applications is atomically packaged and executed. Any changes requires deployment of the entire application.	Creates application or microservices, and packages it in a container Deploys the container to the server. Must manage loading of Docker components and any orchestration/ communication among containers.	Installs or clones an existing OS, and packages the entire OS in a virtual server image and deploys to the server. Developer must stop/stop the entire virtual server.	Developer manually installs middleware and services on dedicated hardware.