Topics for new Software Developer

1. Basic utilities of computing (Linux based OS, Linux Commmands(Bash))
2. Basics of computer programming (How High Level Programming language code converts in binary code and how it is executed?) -- can be excluded if all the cadidates are from Computer Science

3. Go Programming Language:
 a. Introduction to GO (Features, Resources to learn Go, Go Playground)
 b. Setting up Go Environment (GOPATH and GOROOT set up, package structure, breif dependency management) -- Go modules will be covered later
 c. Variables., Constants and their DataTypes in Golang
 d. Arrays and Slices
 e. Maps and Structs
 f. Control Statements (if-else, Looping, switch)
 g. Defer, Panic and Recover
 h. Pointers
 i. Functions
 j. Interfaces
 k. Concenpt of Concurrency
 l. Goroutines
 m. Channels

4. Client-Server Model and Network Protocols:
 a. What is Client-Server Model?
 b. What is an Internet? How do systems interact?
 c. What is a Protocol? What are Network Protocols (IP/TCP/HTTP)?
 d. Importance of Network Protocols in Client-Server Model

5. Datastores:
 a. Different type of Datastores (SQL, NoSQL)
 b. How are they different and when to use what sort of data stores?
 c. How does a Server(In Client-Server) interacts with Datastore?

6. HTTP and HTTPS Protocols:
 a. What is HTTP? Different HTTP Verbs? How are they linked to, the way we access/modify data?
 b. HTTP Request and Response Standards
 c. Implementation of HTTP Server in Golang
 d. GO public HTTP libraries (Gin/gonic)
 e. Synchronous communication

7. Virutalization:
 a. What are Virtual Machines? How are resources shared across different Applications?
 b. Downsides of Virtual Machines? -- Why is it resource-intensive to create VM?
 c. What is a Container? How is it different from VM? Why and How is it easy to create a container compared to Virtual Machine?
 d. What is Docker? What is container orchestration? How Docker solves the problem of VM's heavy lifting?
 e. Build Once -- Run anywhere, anytime Pattern of Software Development

8. Monolithic Servers v/s Microservices:
 a. HTTP server built in above section are monolithic. How can they fail? How hard is it to maintain them?
 b. Logical breakdown of Monolithic Servers in to Microservices (eg: database handlers)
 c. Problems that arise with Distributed System(microservices pattern) -- just a jist, will talk more in detailed section of DS
 d. Synchronous Communication among microservices using Protocols like (HTTP, GRPC, Protobuf e.t.c)

9. Asynchronous Communication among microservices using Message Queue Tools:
 a. What is Asynchronous Communication?
 b. Importance of Asynchronous Communication -- Publish and Subscribe Model
 c. How do we achieve the communication between 2 microservices that are working at different frequency?
 d. Message Queues -- Kafka

10. Docker Commands:
 a. Docker commands to build, run, deploy containers in Docker
 b. Docker registry
 c. Docker Images
 e. Build few containers, using golang and deploy them in Docker

11. Kubernetes:
 a. What is Kubernetes?
 b. Kubernetes Components? (Pods, Deployments, Services, Replica-Sets, ConfigMaps, Secrets e.t.c)
 c. Kubernetes Architecture? (How does above components form K8 Architecture?)
 d. Kubectl set up and basic commands
 e. Kubernetes Configuration File (Yaml) -- Later we move to Helm Chart
 f. Kubernetes Namespaces
 g. Kubernetes Services

12. Developing, Building, Deploying and Maintaining stack of Microservices -- A modern SE playbook