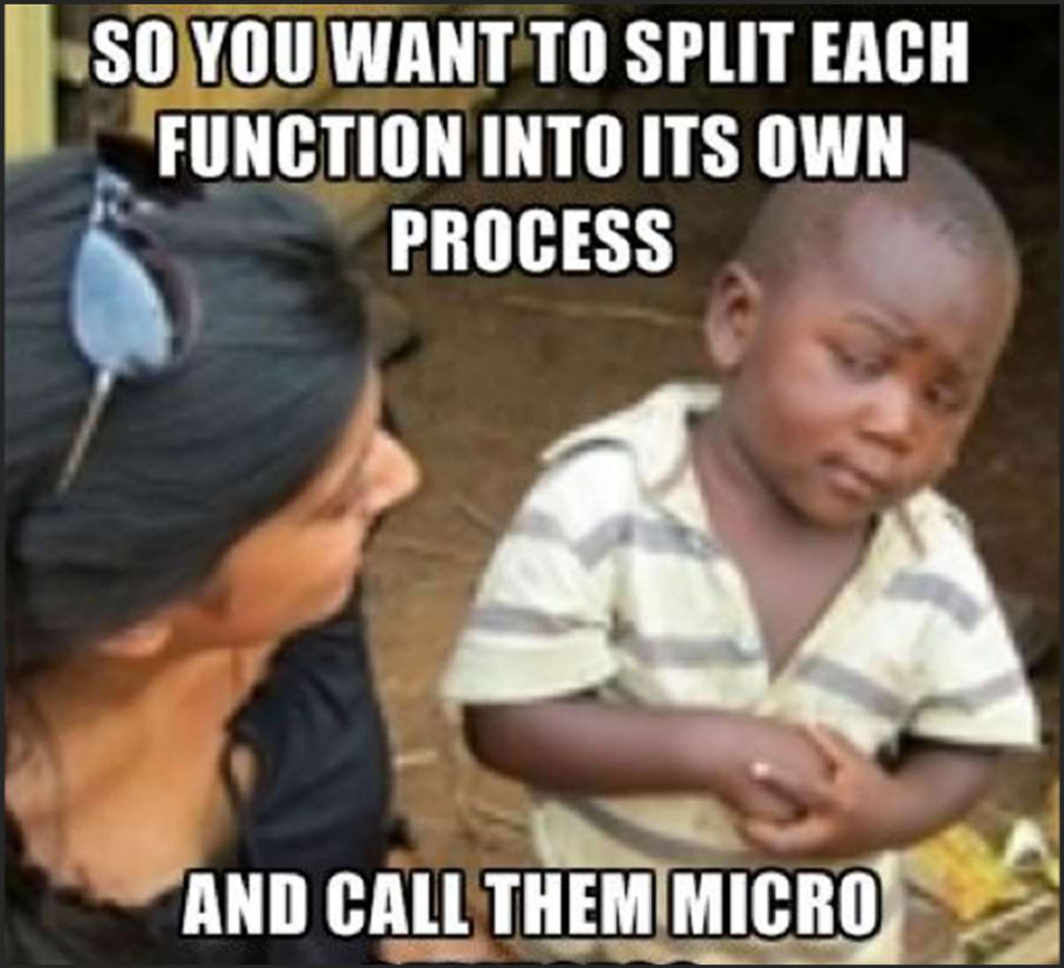
MICROSERVICES?



MICROSERVICES & SPRING CLOUD

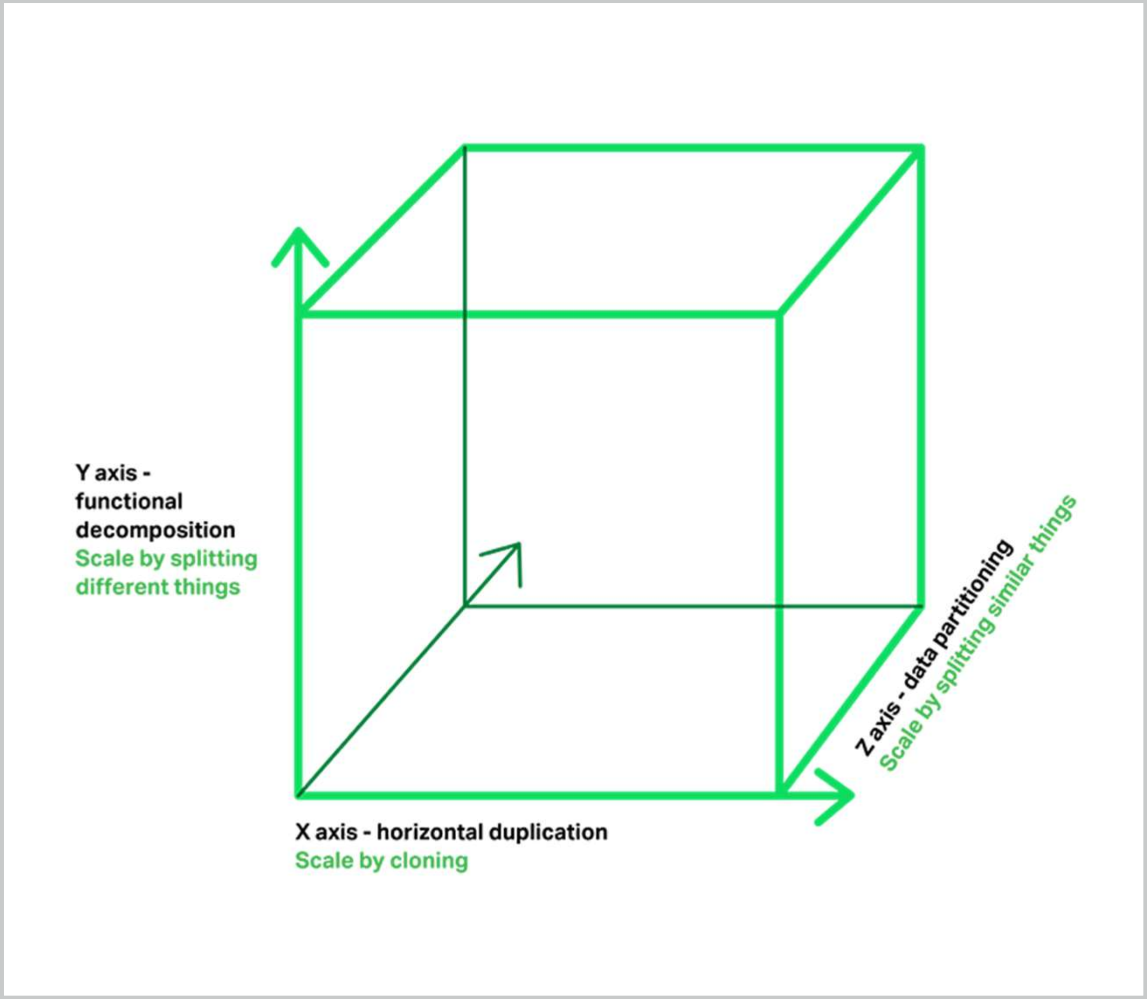
Sujoy Acharya

AGENDA

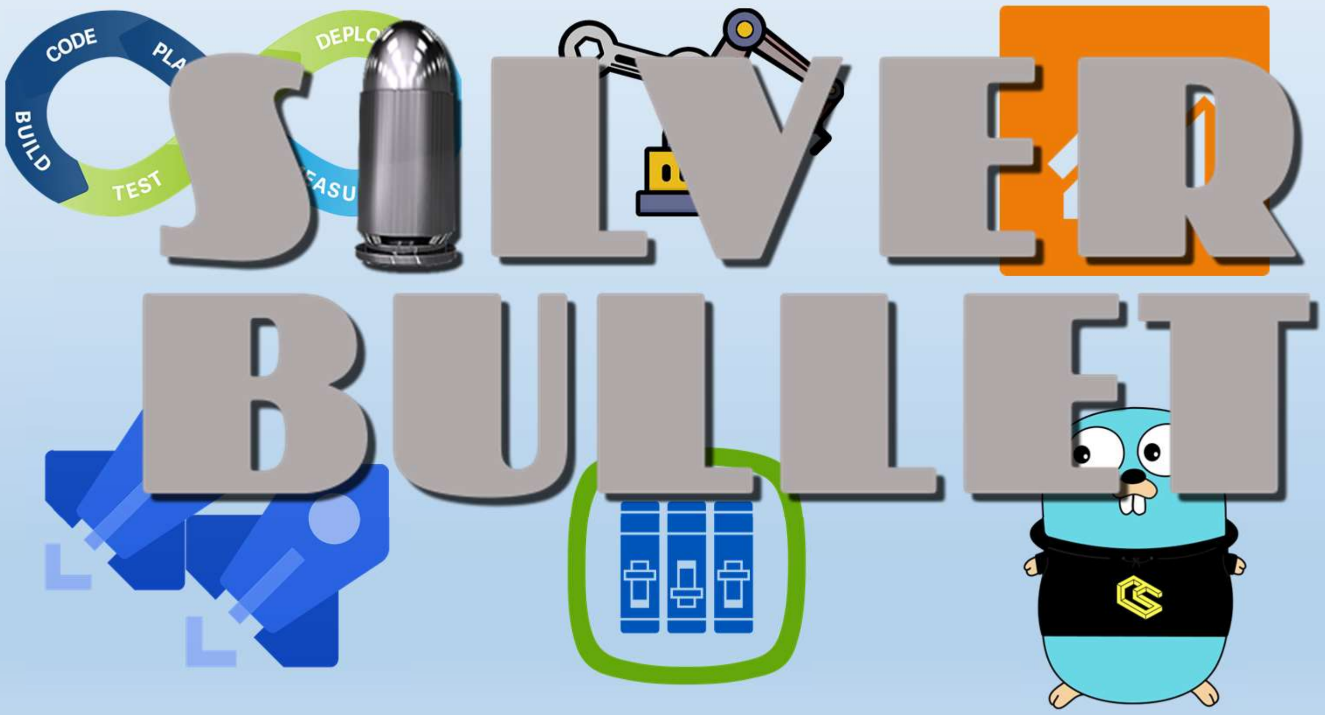
- MICROSERVICES: *RECAP*
- WHY SPRING CLOUD?
  - GATEWAY
  - CONFIG
  - REGISTRY
  - LOAD BALANCER
- EXAMPLES

MICROSERVICES: RECAP

- Structuring an app as a collection of services
  - Modularity
    - Maintainable
    - Testable
    - Loosely coupled
    - Independently deployable


Y axis - functional decomposition
Scale by splitting different things
Z axis - data partitioning
Scale by splitting similar things
X axis - horizontal duplication
Scale by cloning

Microservices: Benefits

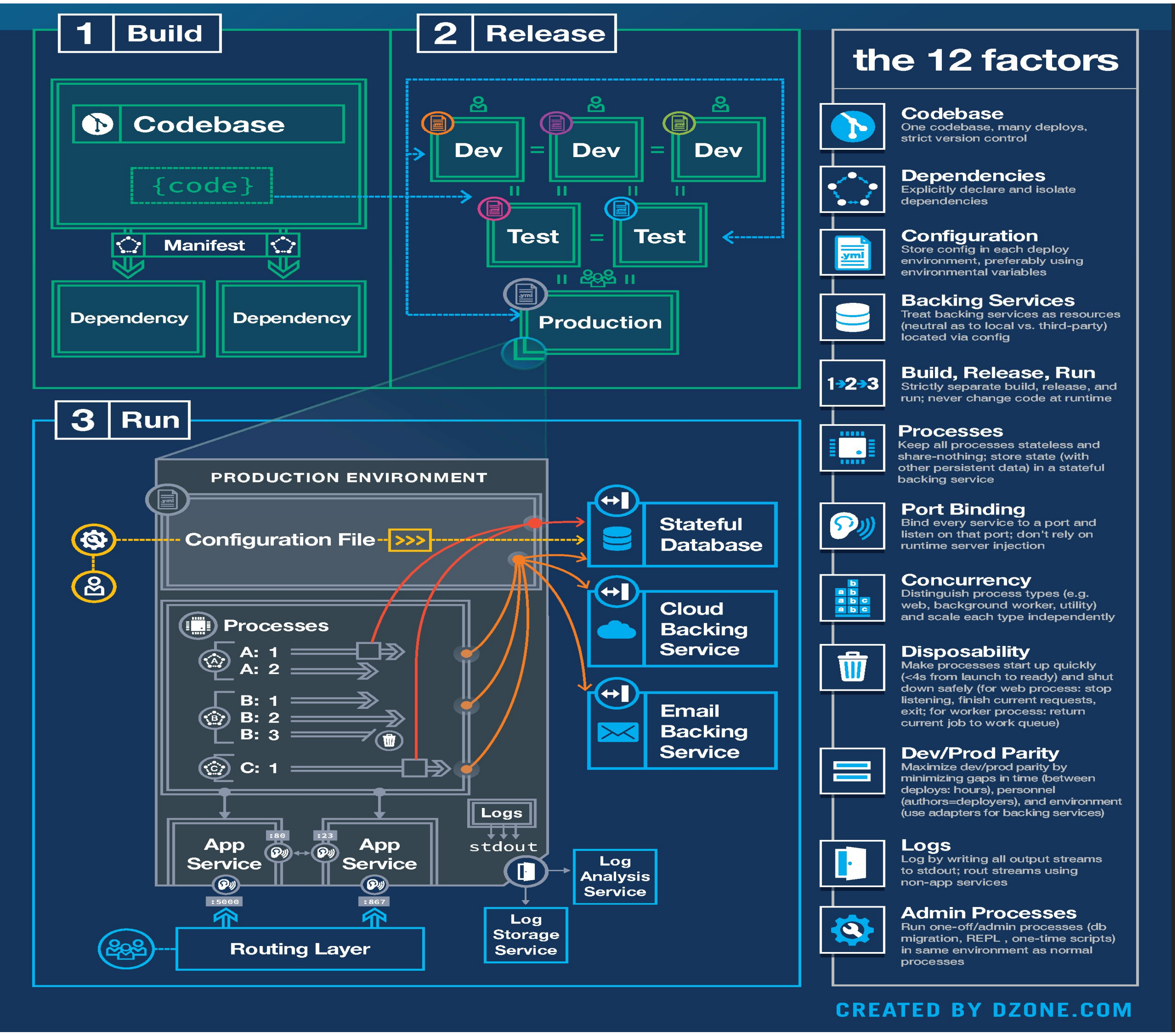


- CICD
- Maintainability
- Scalable
- Deployable
- Fault Boundary
- New Technology

Microservices: Drawbacks

- Decomposition
- Complexity
  - Remote service availability and latency
  - Distributed transaction
  - Distributed query
  - Operational complexity – K8s, docker swarm, PCF
- Multi-service feature deployment
  - Coordination
  - Prioritization

12-FACTOR APPS?



the 12 factors

Codebase
One codebase, many deploys, strict version control

Dependencies
Explicitly declare and isolate dependencies

Configuration
Store config in each deploy environment, preferably using environmental variables

Backing Services
Treat backing services as resources (neutral as to local vs. third-party) located via config

Build, Release, Run
Strictly separate build, release, and run; never change code at runtime

Processes
Keep all processes stateless and share-nothing; store state (with other persistent data) in a stateful backing service

Port Binding
Bind every service to a port and listen on that port; don't rely on runtime server injection

Concurrency
Distinguish process types (e.g. web, background worker, utility) and scale each type independently

Disposability
Make processes start up quickly (<4s from launch to ready) and shut down safely (for web process: stop listening, finish current requests, exit; for worker process: return current job to work queue)

Dev/Prod Parity
Maximize dev/prod parity by minimizing gaps in time (between deploys: hours), personnel (authors=deployers), and environment (use adapters for backing services)

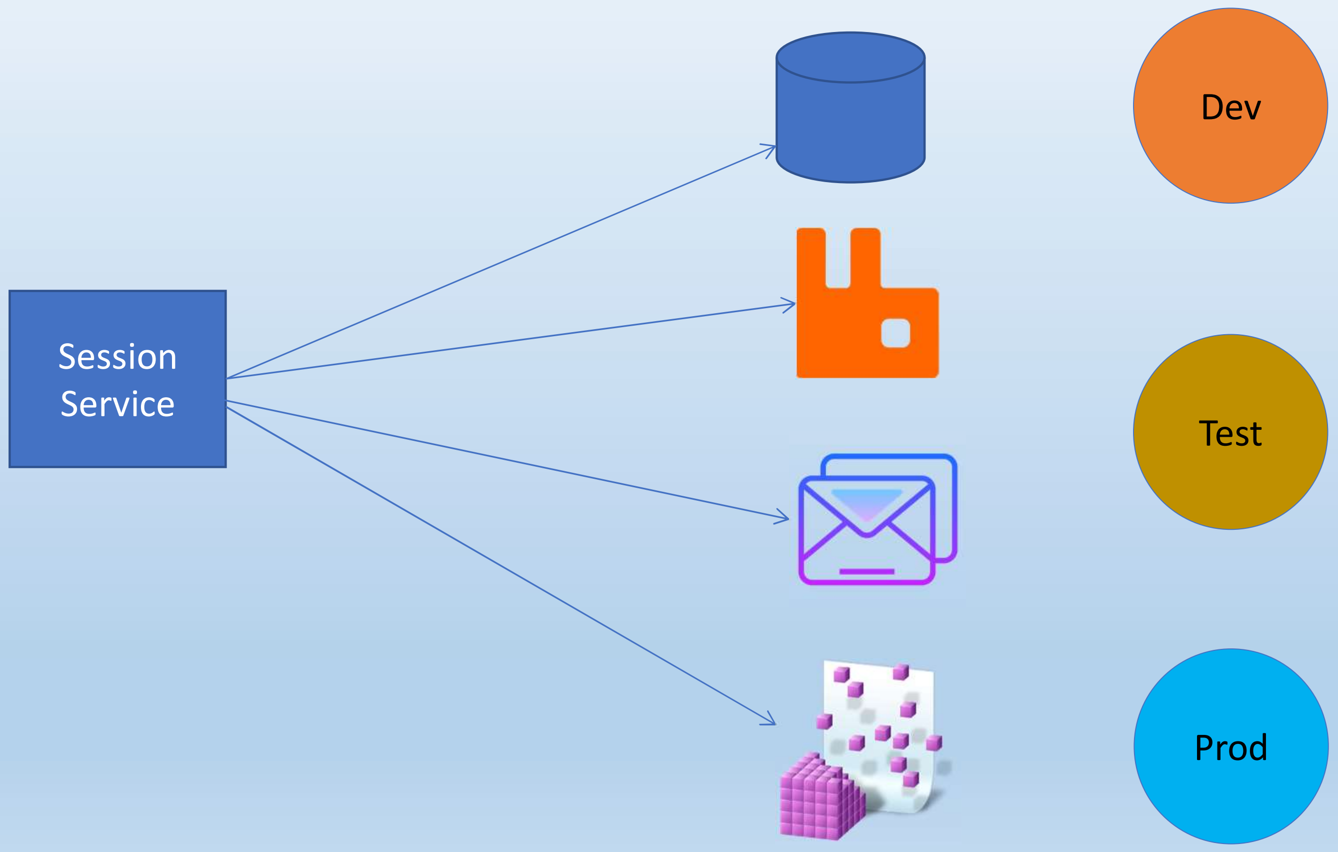
Logs
Log by writing all output streams to stdout; rout streams using non-app services

Admin Processes
Run one-off/admin processes (db migration, REPL , one-time scripts) in same environment as normal processes

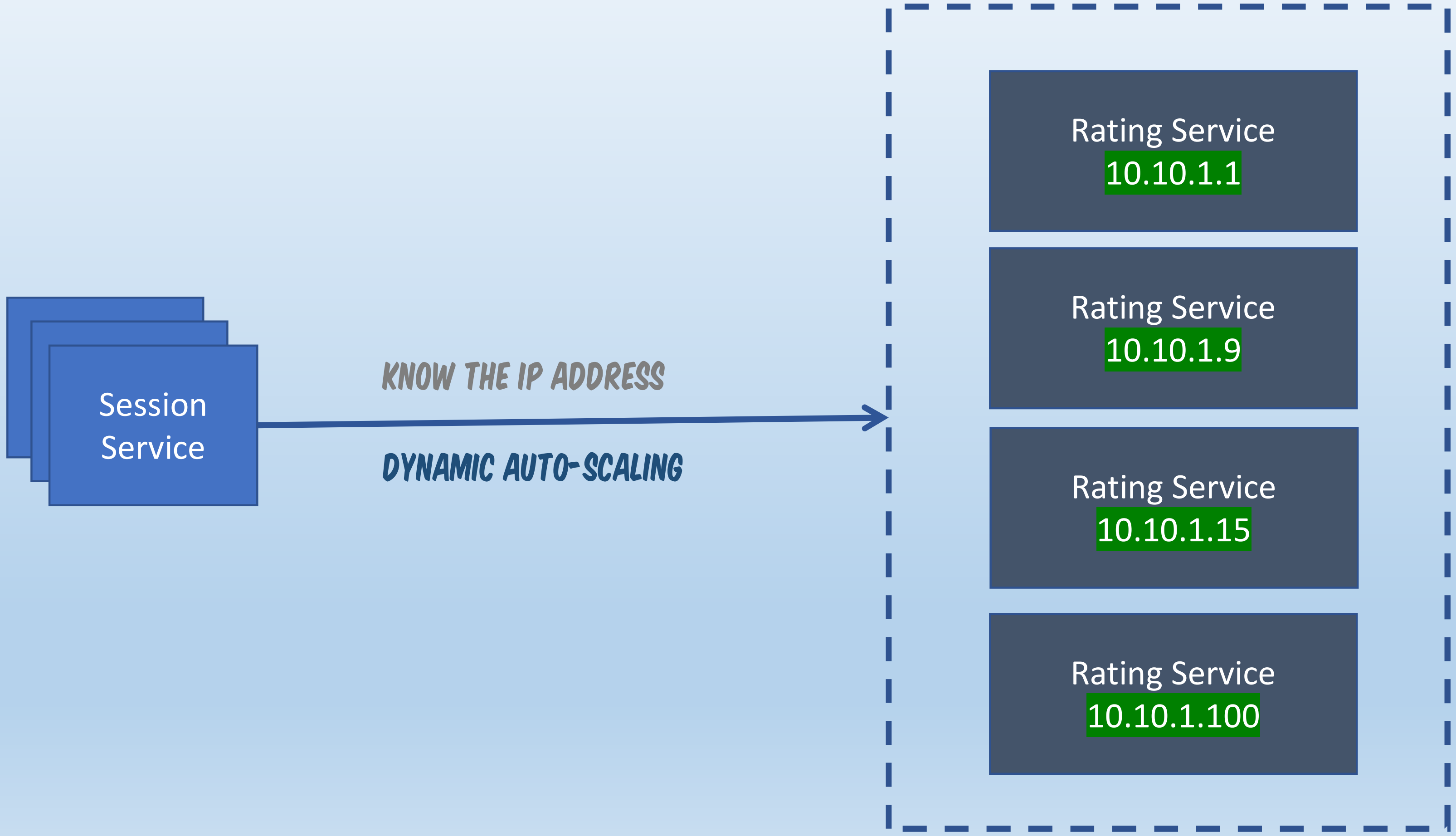
CREATED BY DZONE.COM

CONCERNS?
COMMON

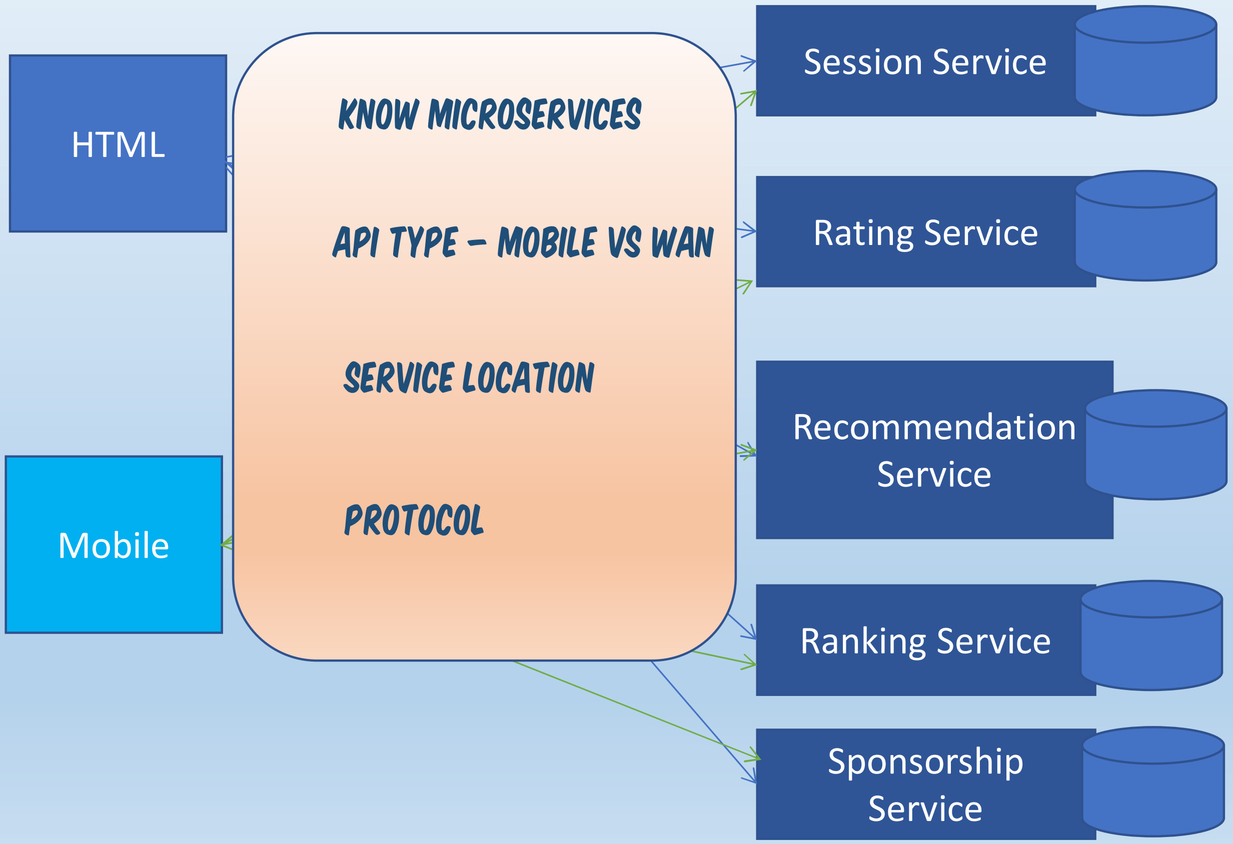
DEPLOYING MICROSERVICES



DISCOVERING SERVICES



INVOKING MICROSERVICES



SPRING CL

- Common concerns
- Tools for building

FEATURES

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Load balancing
- Circuit Breakers
- Distributed messaging
- Distributed tracing
- Data Streaming

CLOUD CONFIG

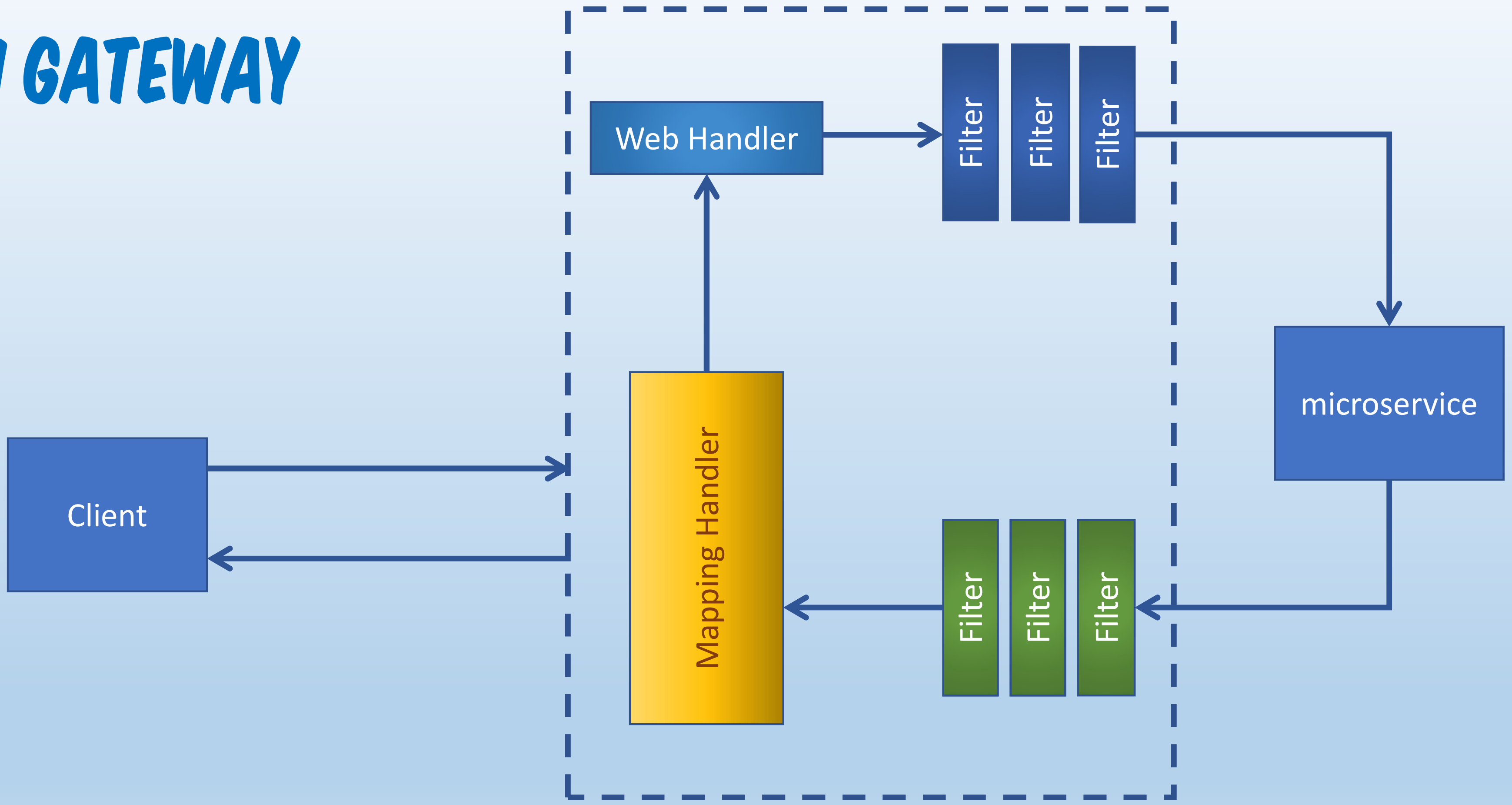
EUREKA AND CONSUL

API GATEWAY

RIBBON AND EUREKA

HYSTRIX, RESILIENCE4J, SPRING RETRY, SENTINEL

API GATEWAY


Web Handler
Filter
Filter
Filter
microservice
Client
Mapping Handler
Filter
Filter
Filter

THANK YOU

References

- 1. 12factor image - https://dz2cdn3.dzone.com/storage/rc-covers/3769958-12-factor-app.png