Microservices Maturity Model 1.0.0 RC1

Microservices Maturity Model	Maturity Levels		
	L1	L2	L3
Characteristics	(Initial)	(Defined)	(Optimized)
Componentization via Services			
Communication via out of process web services	✓		
Independent deployment	✓		
All processes can be deployed together		✓	
Organized around Business Capabilities			
Organization needs for core function	✓		
Organized around capabilities because of Conway's law		✓	
Bounded context with defined context map		✓	
Changes do not cross teams			✓
Products not Projects			
Product is linked to business capability	✓		
Development team runs product end to end			✓
Team owns product over full lifetime (you build it, you run it)			✓
Smart Endpoints and Dumb Pipes			
Decoupled with high cohesion	✓		
Requests are received using RESTish protocols	✓		
Service instance per VM		✓	
Services used most often are cached		✓	
Message queue in place			✓
Service registry			✓
Serverless deployment			✓
Self registration			✓
3rd party registration			✓
Service instance per container			✓
Server-side discovery feature in place			✓
Client-side discovery feature in place			✓
API gateway with caching feature in place			✓
Decentralized Governance			
Useful libraries are shared with other teams using dependency management		✓	
Internal open source model in place for sharing tools and libraries			✓
Service can be created using varied programming languages			✓
Decentralized Data Management			
Uses DB schema and shares DB with other applications	✓		
Has own database		✓	
Allows for compensating transations			✓
Infrastructure Automation			
Continuous integration in place (CI)	✓		
Continuous delivery in place (CD)	✓		
Repository format and branching support automated CI process	✓		
Technical debt detection automated and part of CI		✓	
Unit tests are automated and part of CI		✓	
UI tests are automated and part of CI/CD			✓
Design for Failure			
Real-time monitoring of architectural elements (e.g. requests per second)		✓	
Real-time monitoring of business metrics (e.g. applications per minute)		✓	
Chassis (externalized configuration, logging, and health checks)		✓	
Service failures are introduced into production and tested			✓
Automated testing in production			✓
Evolutionary Design			
Seperation of high vs low change code	✓		
Tooling to allow for frequent, well-controlled changes to software		✓	
High isolation level. Can be scrapped and replaced			✓

"Microservices: a definition of this new architectural term" Martin Fowler and James Lewis
http://martinfowler.com/articles/microservices.html

"Defining the Business Capability - A Cheat Sheet" William Ulrich
http://www.bainstitute.org/resources/articles/defining-business-capability-cheat-sheet

"Pattern: Microservices Architecture" Chris Richardson
http://microservices.io/patterns/microservices.html