Stage	Activity	Performance level 0 Low performers	Performance level 1 Medium performers	Performance level 2 High performers	Performance level 3 Elite performers
Plan	Approach	Project approach with big specification up-front (“waterfall’”)	Iterative project approach (agile/ ”Scrum” principles & techniques)	Iterative product approach, driven by product backlog	Flow of changes (Kanban-style)
	Team	Separate build and maintenance team	technology knowledge divided over different teams (change request driven)	all required knowledge in the team	generalizing specialists
	Work visualization	Nothing	only planned work	only own work	all work is visualized
Code	Quality	Nothing	Coding guidelines	Manual reviews	Automatic code scans
	Versioning	Nothing	Just versioning	Branching per release	Trunk based development
	Security	Nothing	secure coding guidelines	External library scans	code scans (SAST tool)
Build	Approach	Manual build	Scheduled nightly	scheduled every hour	upon commit
	Breakers	compilation errors	failing unit tests	code quality scans	Security scans
Test	Approach	Manual testing in UAT	Automated functional tests (unit, integration)	Automated non-functional tests (stress, load)	Chaos injection in production
	Responsibility	Business is responsible	Business and IT all do their part of the tests (overlap)	Business and IT do their part of the tests (no overlap)	Shared responsibility: business decides what to test, IT decides how to test it
	Security	Nothing	Penetration testing before going to production	Recurring penetration testing	DAST tool continuously scans behavior or running application and reports vulnerabilities

Stage	Activity	Performance level 0 Low performers	Performance level 1 Medium performers	Performance level 2 High performers	Performance level 3 Elite performers
Release	Approval	Separate release management team guarding over the planned changes and their impact	Business go-no go meeting	Product owner	Team – 4 eyes principle
	Activate	upon deploy	fixed date, via feature toggles	on demand Via feature toggles	On commit Only with trunk based development
Deploy	Frequency	Quarterly	Every month	Every week	Constant flow Only with Kanban-style plan approach and trunk-based development
	Code	Manual deploy Separate team	Automated build	Automated deploy to test, manual approval for UAT, production	Automated deploy to production Only with Kanban-style plan approach and trunk-based development
	Infrastructure	Know your colleague: call the infra guy to do the changes	Get infrastructure changes via service request	Changes to infrastructure are done via self service	Infrastructure as code, part of your source code repository
Operate	Team	Segregation of duties	closer collaboration between dev and ops team	shared responsibility between dev and ops team	you build it, you run it E2E team responsibility
	Availability	Only 1 production instance	Cold standby	Hot standby	Load balancing and failover
	Capacity	No capacity management in place	Fixed capacity, based on historic usage and capacity metrics	Elastic (manually sized) capacity, based on historic usage and capacity metrics	Automated capacity management based on usage metrics and feedback
Monitor	Approach	Nothing	Information radiators followed up by a separate team	Automatic escallation to team members	Self-healing & self-learning system