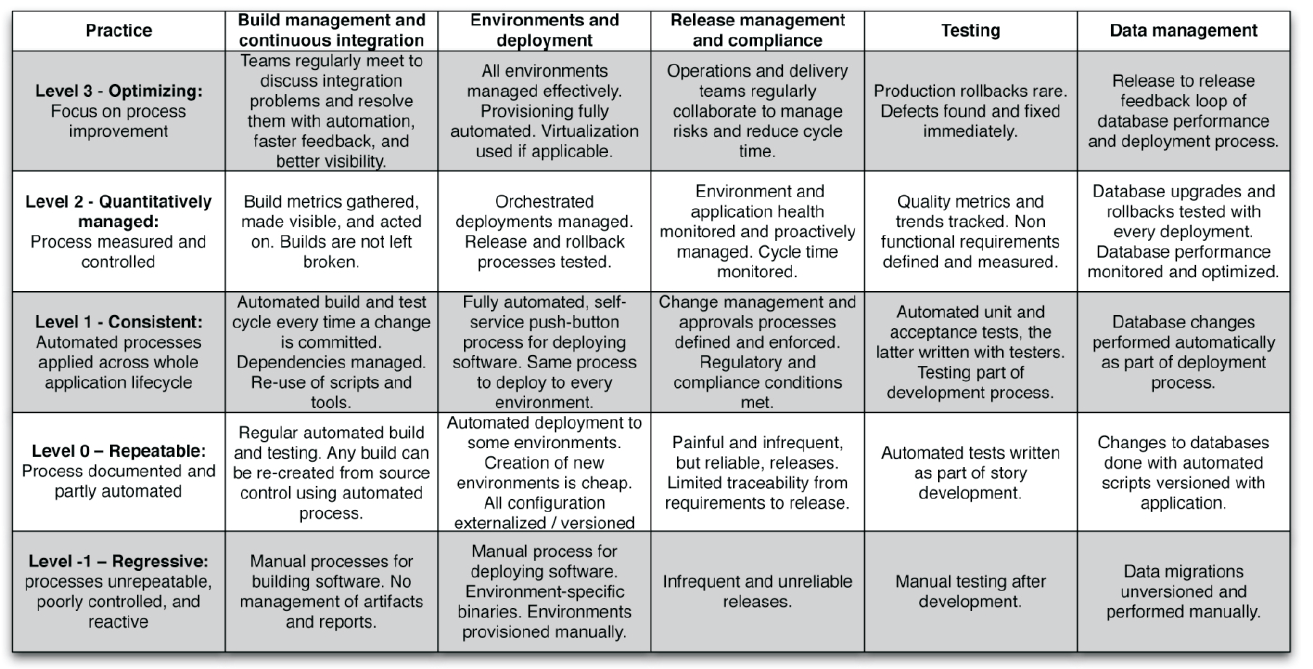
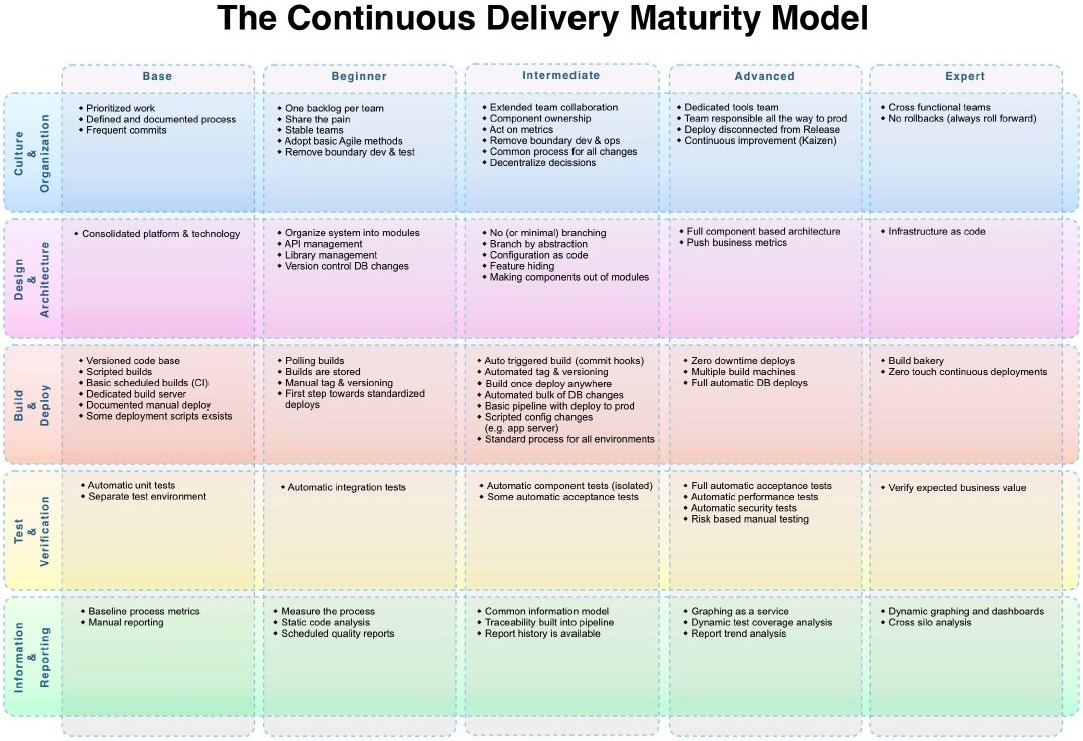
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Adoption Level:** | **Developing** | **Automating** | **Optimizing** | **Inspiring** |
| **Primary Focus** | * Fully scripted deployment * Compliance with T&D, division, ADG policies * Customized tooling to ensure smooth deployment in dev/qa/uat/prod environments | * One push button deployment * Database continuous integration * Building/Automating its own deployment pipeline * Creating automated reusable test suits through collaboration between developer and QA | * Creating clear quality gate for each deployment stage * Automated deployment to production after passing quality gate * Optimizing deployment pipeline * Architecture changes with continuous deployment in consideration | * Zero downtime deployment * Roll forward/back is automated tested * Visualized real-time monitoring dashboard of deployment activities for stakeholder * Measureable metrics to help dev/user make IT investment decision |
| **Commit Stage** | * Codes are maintained in firm approved version control repository * Codes are peer reviewed before checkin * Develop on release branch and merge after release * Sufficient unit testing to ensure code quality | * Database changes are scripted and put into version control * Master/trunk are always releasable (no hard requirement on release branch) | * Every code checkin that pass deployment pipeline is releasable * Deployment pipeline automatically rejects changes that break quality gates | * Cross function team work together to define work item and how to realize business values * Developer productivity is measured via statistics |
| **Build Stage** | * Train adoption compliance with T&D policy * Automated unit tests * Clearly defined quality gates for code coverage ratio, security analysis | * Apply database continuous integration * Developers build automated functional tests | * Build once and deployable anywhere * Standard build process across teams, environments * Configuration management as first-class citizens available through entire lifecycle | * Automatic collected build metrics to visualize delivery quality trending and bottleneck * Create deployable container * Automated tests at all level |
| **Release Stage** | * Train adoption compliance with T&D policy * Manual environment provisioning | * Close-to-production environment is provisioned from codes | * Automated production-like environment provisioning across deployment pipeline * Fully automated regression testing suits * Fully automated performance testing suits | * Decouple release from deployment with architecture support * Non functional requirements are defined and automatically measured |
| **Deploy Stage** | * Customized scripts for automated deployment, but still a large portion of deployment activities are done manually * None or limited automated functional tests * None or limited automated roll backward/forward capabilities | * Fully automated deployment using scripting * QA team collaborates with development team to build automated reusable functional testing suits | * QA team focus on building automated reusable testing suits * None or limited manual verification steps required post turnover as majority of verification are automated | * Feature gate to enable/disable functionalities * Run time configuration to change application behavior without down time * Zero downtime deployment using dark launch, blue-green deployment * Systems are forward/backward compatible by design |





microservice

* separately deployable and separately deployed
* SOLID principle
* dependencies create larger batches

feature switch to enable parallel deployment for changes that involve multiple components

Branch by Abstraction

virtualize your pipeline

1. branch by abstraction
2. feature switches
3. piece meal dark deployment

deploy abstractly - truly cloudy, no care about machines, ip, dns, certificates

all or nothing deployment is anti-pattern

Test in Production (TiP)

environment is provisioned from code

Deployment goals

* fully scripted deployment
* push button deployment
* push button deployment to production
* automated deployment after tests pass, 0 downtime deployment, release and rollback is tested

factors that require evolve: source control, build process, deployment, testing & qa, visibility,

visibility - stakeholders have dashboards with real-time product and dependency stats, cross team data mining and analysis

**source code repository**

github

perforce

svn

bitbucket

mercurial

helix

**build tool**

maven

msbuild

gradle

ant

msant

scala/.net build tool

**CI**

jenkins

bamboo

teamcity

go

**Repository management**

Nexus

**Testing**

selenium

cucumber

junit

jmeter

appium

testcomplete

**Deployment**

go

codedeploy

ssh

rapiddeploy

rundeckxl deploy

ultracode deploy

**Config/provisioning**

chef

puppet

ansible

salt

vagrant

**containerization/virtualization**

docker

mesos

kubermeters

cloud/laas/paas

aws

azure

heroku

apprenda

**release management**

xl release

tcm

bmc release process

**collaboration**

jira, slack, trello

servicenow

**BI/Monitoring**

kibana

new relic

**logging**

splunk

logstash

elasticsearch

**security**

snort

tripwire

cyberark