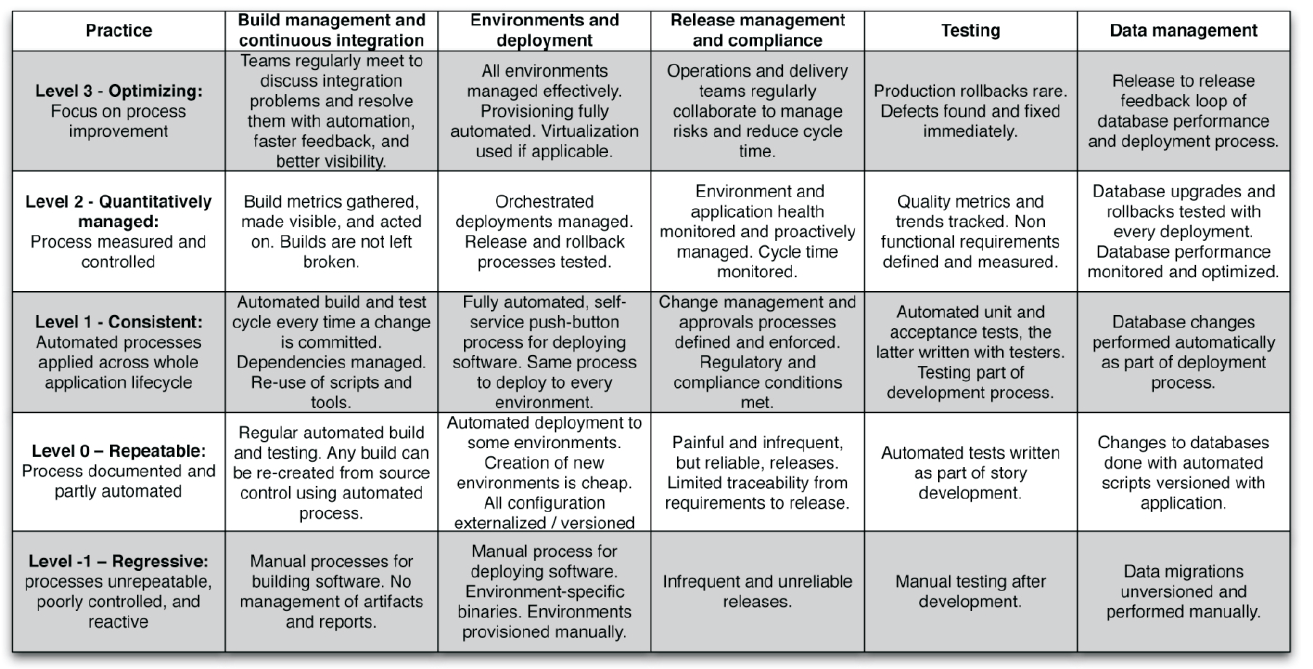
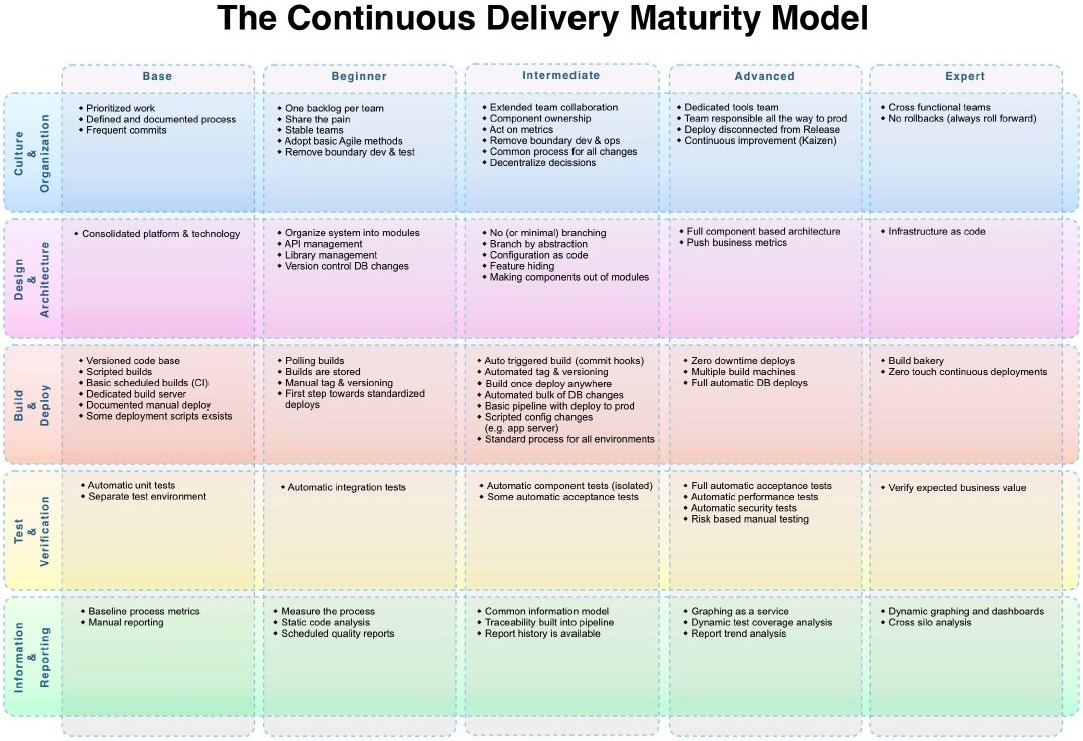
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Adoption Level:** | **Developing** | **Automating** | **Optimizing** | **Inspiring** |
| **Primary Focus** | * Fully scripted deployment * Compliance with T&D, division, ADG policies * Customized tooling to ensure smooth deployment in dev/qa/uat/prod 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 after passing quality gate * Optimizing deployment pipeline * Architecture changes with continuous deployment in mind | * Zero downtime deployment * Roll forward/back is automated tested * Visualized 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 * Sufficient unit testing to ensure code quality | * Database changes are scripted and put into version control * Apply database continuous integration |  |  |
| **Build Stage** | * Train adoption compliance with T&D policy * Automated unit tests * Clearly defined quality gates for code coverage ratio, security analysis |  |  |  |
| **Release Stage** | * Train adoption compliance with T&D policy |  |  |  |
| **Deploy Stage** | * Customized scripts |  |  |  |





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