CI / CD

Continuous Integration / Continuous Delivery

Förr så arbetade varje team för sig, ibland varje utvecklare för sig, och man väntade till sista dagen innan man mergade ihop allt arbete.

Detta ledde givitvis till extrema problem, och allt som oftast fick man en del oönskade sidoeffekter på köpet.

Dessutom tvingades då hela laget sitta tillsammans och "cherry-picka" ihop koden för den skulle fungera.

CI / CD är ett sätt att automatisera alla steg i utvecklingen och releasen av projektet man utvecklar.

Automatiserade pipelines eliminerar mänskliga fel, tillhandahåller en standardiserad feedback på utveckling och skyndar på iterationernas kodbas.

CI innefattar ett centralt repository som alla utvecklare arbetar emot, som varje dag kan ta emot hundratals pushes, merges och rebases.

Med CI så varje kod förändring triggar den automatiserade bygg och test processen. I olika verktyg som Jenkins, Azure, AWS, Gitlab CI m.fl. kan man följa processen där man laddar hem test miljöer som är exakt likadana som produktions miljön.

Processen innefattar att tester körs, artifacts skapas och man får fullständig feedback på vad som skett.

Denna process får aldrig ta mer än max 10 minuter.

Tar den mer än 10 minuter, så blir risken betydligt högre att utvecklare tummar på test biten.

Kanske skippar man ett test för en enkel ny feature, eller så väljer man att tvångstrigga deployment innan testerna har körts.

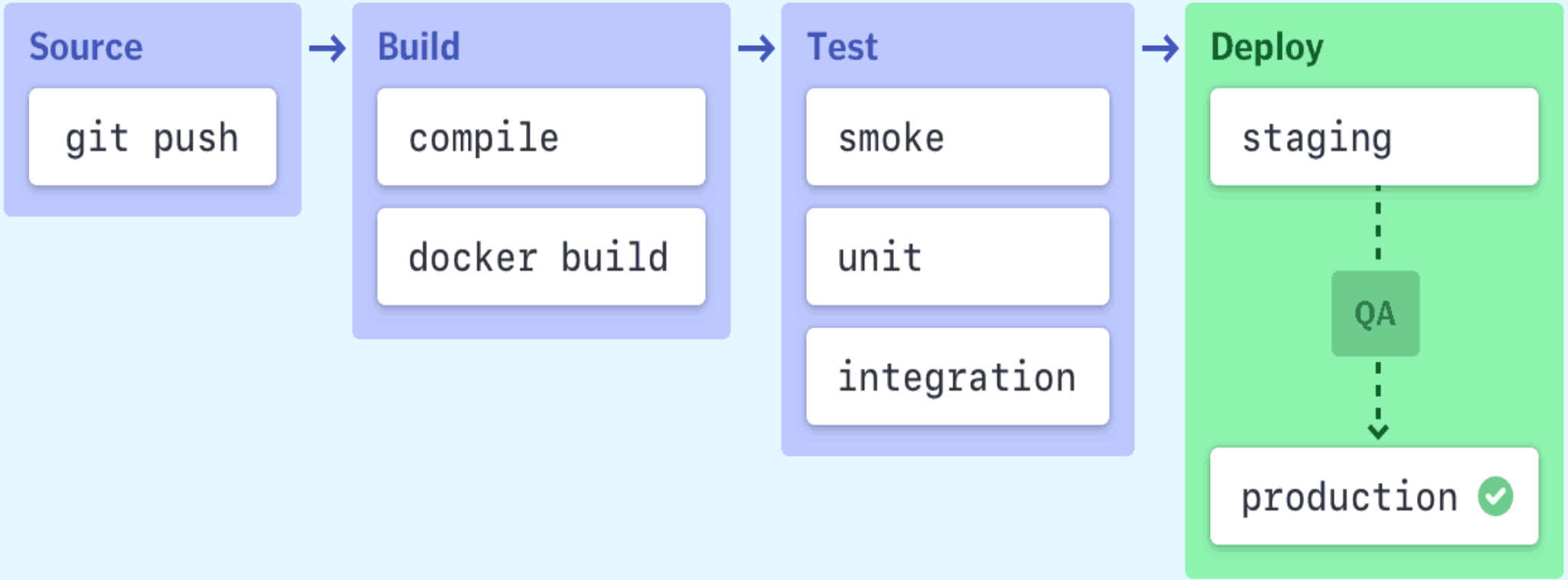
Dessutom om utvecklingslaget finns spritt runt flera tidszoner har det visat sig att utvecklingen stagnerar, då utvecklarna måste invänta tidigare commits test-runs.

CD tillhandahåller pipe:n från artifact repository ner till produktionsmiljön.

Det kan innehålla en, till flera steg, men faller på att artifact repository "pingar" en server om att något nytt har dykt upp.

Servern i sin tur tar då och kopplar upp sig emot repositoriet, börjar ladda hem den nya artifacten.

Den nya artifacten laddas alltid hem på en "staging" miljö, dvs en produktionsmiljö, men som inte kan nås av användaren förräns den blir live.


Source
git push
Build
compile
docker build
Test
smoke
unit
integration
Deploy
staging
QA
production

CONTAINERIZATION

Eftersom det sker multipla git pushes varje dag mot repositories, så krävs det minimalt med belastning på servern.

Det är detta Docker, Docker-Compose och Kubernetes är till för.

Man spinner helt enkelt upp en container som simulerar enbart miljön som behövs. På detta sätt har vi kopplat bort hårdvara och mjukvara vilket gör att man kan deploya sin produkt på vilekt system som helst som kan stöda den utvecklingsmiljön man använder.

Docker är en "motor" som läser en Dockerfile och skapar en minimal VM med just det innehållet.

På detta sätt används minimalt med utrymme, och när man inte behöver den mer, så stänger man av den och eventuellt slänger bort den ifall man inte behöver den miljön mer.

Docker är ett väldigt vanligt verktyg i utvecklingen idag, och tillägget Docker-Compose är egentligen bara ett sätt att köra multipla Dockerfile i ett och samma kommando.

Man brukar säga det finns fyra steg i datormiljön:

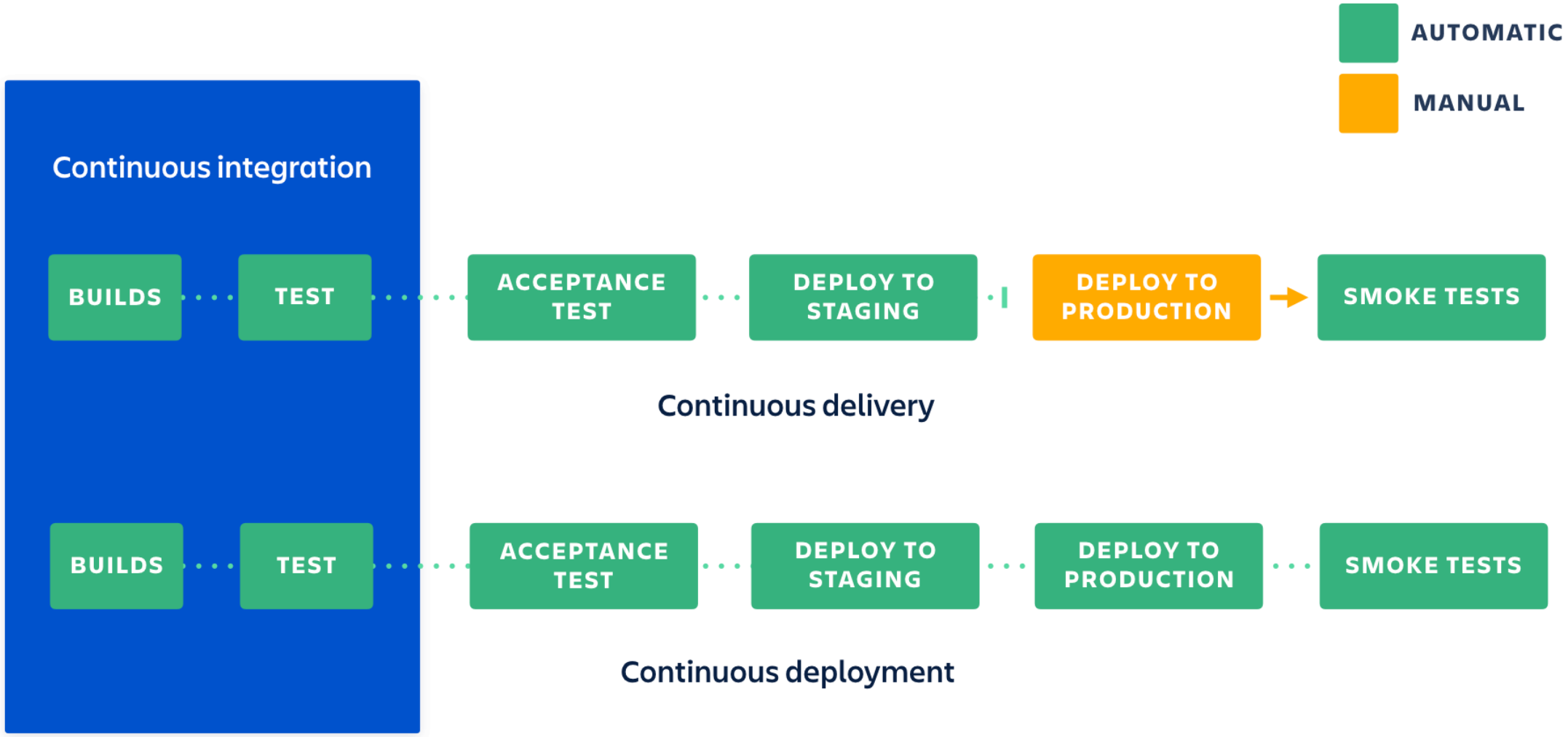
- Appar / Program
- Virtual Machine Engine
- Mjukvara
- Hårdvara


Build, Ship, Run, Any App Anywhere
From Dev
To Ops
Any App
Microsoft
ASP.NET
.NET
Java
MORE
CONTAINERIZATION ENGINE
Any OS
Windows
Linux
Anywhere
Physical
Virtual
Cloud

CD eller CD?????

Continuous Delivery eller Continuous Deployment.

Vad är skillnaden?


AUTOMATIC
MANUAL
Continuous integration
BUILDS
TEST
ACCEPTANCE TEST
DEPLOY TO STAGING
DEPLOY TO PRODUCTION
SMOKE TESTS
Continuous delivery
BUILDS
TEST
ACCEPTANCE TEST
DEPLOY TO STAGING
DEPLOY TO PRODUCTION
SMOKE TESTS
Continuous deployment

Kostnad / Vinst med CI

Förlust

- Skriva tester
- Dedikerad server
- Måste merga sitt arbete minst en gång / dag

Vinst

- Mindre buggar
- Alla integration problem är lösta
- Eventuella fel syns direkt
- Test kostnader sjunker

Kostnad / Vinst med CD (Delivery)

Förlust

- Skriva tester med acceptence kritier
- Deployments måste vara automatiska
- Versionshantera koden

Vinst

- Teamet måste inte spendera dagar i förberedelse för release
- Releaser kan ske snabbare, vilket ger snabbare feedback
- Mindre press för små förändringar

Kostnad / Vinst med CD (Deployment)

Förlust

- Extremt bra tester
- Snabb dokumentation process
- Feature flags blir viktiga

Vinst

- Ännu snabbare utveckling
- Mindre risk för buggar och i regel enklare att fixa om man släpper små batches.
- Kunderna ser förändringar varje dag eller vecka istället för årsvis