Kogebogsprincippet for Changes og SOPs

Formål

For at man kan lave gode Changes og SOP'er (Standard Operating Procedures) er det ligesom at være en kok der skal lære at lave god mad, eller måske den matra man kører indenfor militæret - gentagelse, gentagelse, igen, igen..

Målet er at ende med en god opskrift, så vi kan "lave god mad" hvergang 🙂 Det kalder jeg "Cookbook" princippet

Lad os starte med de systemer vi normalt har at arbejde med:

System	Formål	Hvem kan deploye her	Gode/dårlige ting
Development	Grundudvikling, dette kan ofte være udviklerens egen maskine eller viruelle systemer	Developer	Flygtigt miljø med mange ændringer og ingen kontrol Meget flexibelt med mulighed for stor produktivitet
Test	Test af det der er udviklet på Development	Developer	Flygtigt miljø med mulige ukendte ændringer
Stage	Deployment af det der er udviklet på Development og teste på Test	Sysadmin (Evt. Developer ved DevOps)	Kendt miljø under Change kontrol. Få eller ingen ændringer
Production		Sysadmin (Evt. Developer ved DevOps)	Kendt miljø under Change kontrol. Få eller ingen ændringer og skal til enhver tid matche Stage miljøet

Der kan sagtens existere flere miljøer end de 4 ovenstående - f.eks er disse relativt normale:

System	Formål	Hvem kan deploye her	Gode/dårlige ting
UAT	User Acceptance Test	Sysadmin (Evt. Developer ved DevOps)	Kendt miljø under Change kontrol. Få eller ingen ændringer og skal til enhver tid matche Stage miljøet
integration	Test omkring integrationer med andre systemer, f.eks import/export/synkronisering	Sysadmin (Evt. Developer ved DevOps)	Kendt miljø under Change kontrol. Få eller ingen ændringer og skal til enhver tid matche Stage miljøet

DevOps

DevOps og Continues Delivery koncepterne bryder lidt med den traditionelle Udvikler vs. Sysadmin tanke, idet at man ofte antager at Sysadmins blot kan deploye, mens udviklere ikke kan arbejde (overhovedet) i Stage eller Produktionsmiljøer.

Dette er ikke nødvendigvis koliderende tankegange, men man bør overvejer og implementere lidt anderledes regler og godkendelsesprocedurer, da jeg (belært af små 10 år på Sysadmin siden) må erkende:

> Udvikleres synsvinkel på klassiske dyder og procedurer er som regel koncentreret om helt andre aspekter end dem som en Sysadmin eller driftsansvarlig person fokuserer på - for udvikleren har typisk et fokus på at komme hurtigt i mål med nye features og/eller deployments, - somme tider på bekostning af tests og datadiciplin. Giver man udvikleren - eller tvinger han sig adgang - til Produktionsmiljøet kan man som Sysadmin godt anse slaget for tabt.

Ved DevOps eller Continues Delivery bør man evt implementere regler, procedurer eller mere tekniske funderede adgange som godkendelser eller workflows hvor deployment afhænger af success tests i de miljøer der er før det ønskede miljø for deployment, for at forhindre utilsigtet eller forhastet deployment fra udviklerens side.

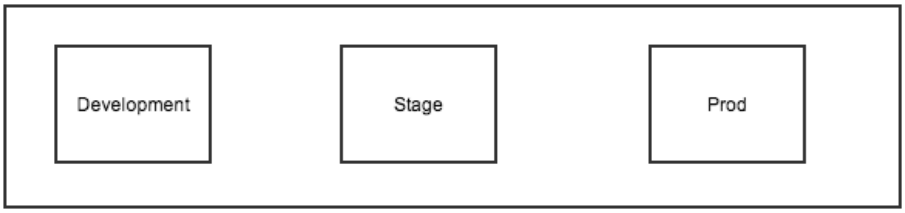
Der er også datasikker- og datafortrolighed at tænke over, når udviklere gives nogen som helst form for adgang til Stage eller Prod, som ofte indeholder personhenførbar eller fortrolig data, både fordi at antallet af personer med tilgang simpelthen udviddes, sekundært fordi udvikleren [potentielt] har mulighed for at udnytte softwaren til at udtrække fortroligt data udenom normale foranstaltninger.

End af udviklernes typiske grunde til at ænske eller begære adgang til Stage eller Produktion er kravet om at kunne debugge eller må logs i realtime, her anbefaler jeg produkter som splunk for at få data (på en kontrolleret måde) ned fra serverne/systemerne til udviklerne.

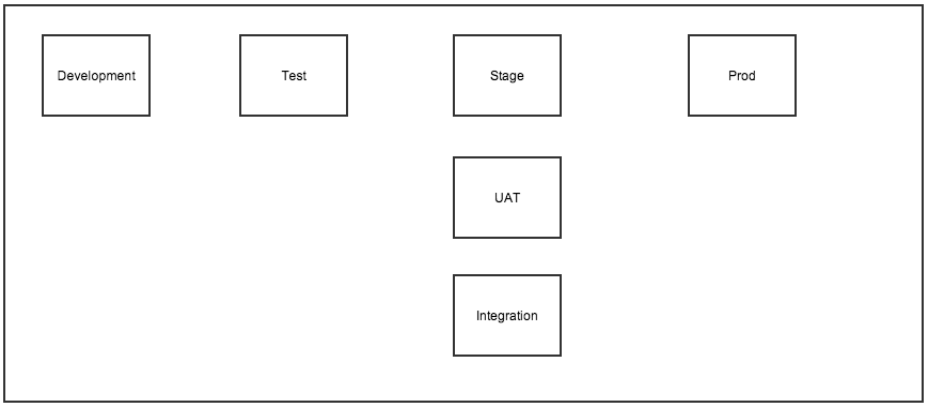
Sekundært er der IT Revisionsmæssigt gode grunde til at funktionsadskille.

Miljøer

Vores absolutte minimum må være:

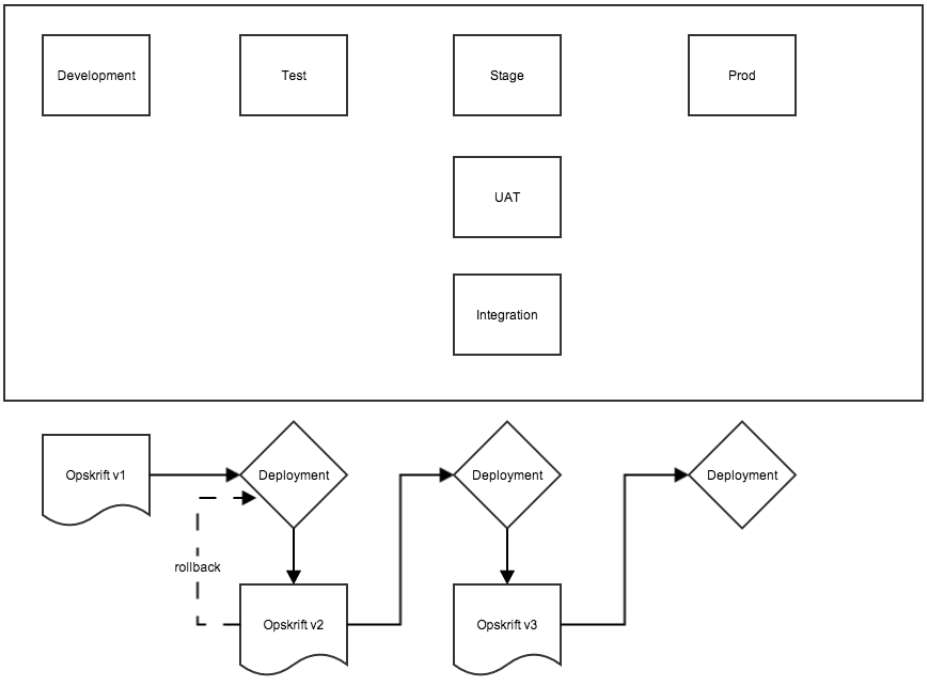


Endnu eller endnu bedre:



Oprettelse af opskrift og gennemførsel

Ligemeget hvor mange systemer eller rettere niveauer der er på vores platform, har vi mulighed for at forbedre vores opskrift op gennem de forskellige niveauer fra Development til Production, og med dagens teknologier indenfor virtualisering hvor vi kan rulle en server eller et system tilbage til et tidligere state på meget kort tid (via snapshot teknologi) - kan man indenfor de forskellige systemer sagtens gentage processen flere gange:



Fra opskrift til SOP

En opskrift i kogebogen er ofte kun fremadrettet, dvs. tanken om at gå tilbage er ikke altid med i opskriften, ofte har vi ved fejl bare rullet tilbage til et snapshot og startet forfra.

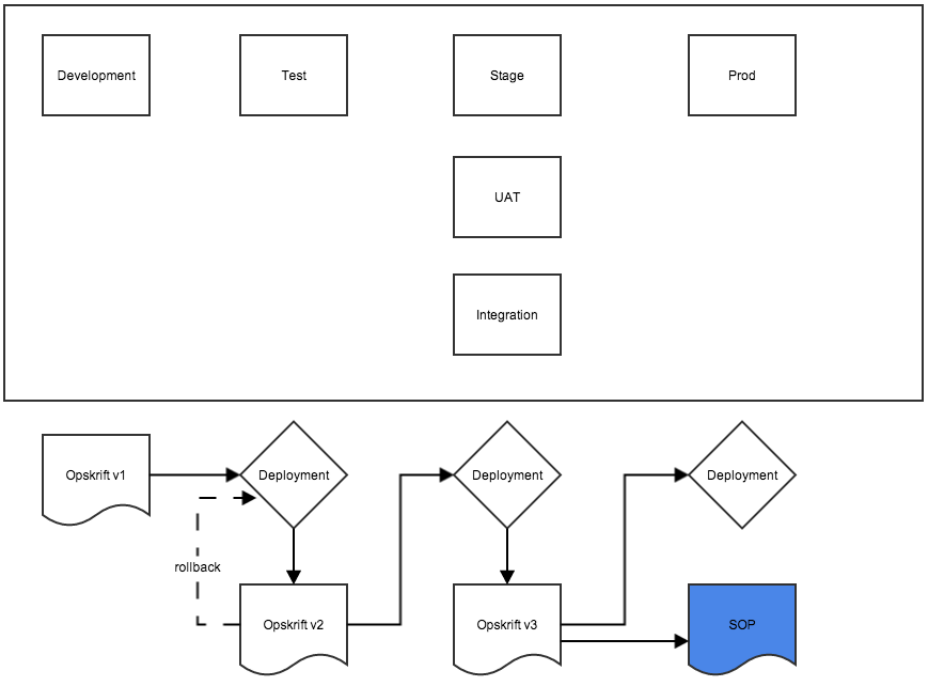
Men der er mange situationer, hvor denne rollback metode ikke er mulig eller ønskelig, specielt i systemer hvor nedetiden skal være minimal, eller der er integration til andre systemer, der ikke tillader rollback, f.eks kunne et fortløbende ID smadres.

Samtidig har opskriften måske mange objecter med navne og bestemte ting for netop denne opskrift.

Det kan derfor være ønskeligt at "rense" opskriften for disse ting og lave en SOP - hvor alle navngivninger er ændret i nomenklateren til placeholders for objekter etc. Det er så også ønskeligt at der med eller i SOP'en følger en forklaring og/eller liste af hvad der skal i disse placeholders når SOP'en bruges (Pre-requisite).

Se Template for en SOP

Det sidste skridt der derfor at SOP'en laves:



Del og Hersk

Del og Hersk er et vigtigt princip for gode changes (og SOPs) - frem for at have en meget stor Change med mange skridt og implementationstyper (Applikation, Firewall, Database, OS) bør man dele Changen op i flere af hinanden enten uafhængige eller sekventielle Changes der hver for sig kan gennemføres. Dette betyder også at et evt. rollback af en Change ikke er så kristisk for den samlede masse af Changes.